НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Комп'ютеризовані системи управління»
спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології»
на тему: «Автоматична система моніторингу та аналізу кількості пасажирів у транспортних засобах»

Виконав:
студент IV курсу, групи ІА-61
Комар Антон Аркадійович _____

Керівник:
Доцент, к.т.н,
Сокульський Олег Євгенович _____

Рецензент:
Доцент кафедри АСОІУ, к.т.н.,
Попенко Володимир Дмитрович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 151 «Автоматизація та комп'ютерно-інтегровані технології»

Освітньо-професійна програма «Комп'ютеризовані системи управління»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ
на дипломний проєкт студенту
Комару Антону Аркадійовичу

1. Тема проєкту «Автоматична система моніторингу та аналізу кількості пасажирів у транспортних засобах», керівник проєкту Сокульський Олег Євгенович, доцент, кандидат технічних наук, затверджені наказом по університету від «7» травня 2020 р. №1081-с

2. Термін подання студентом проєкту 11.06.2020

3. Вихідні дані до проєкту

Мови програмування Python та C#, бібліотеки OpenCV та Asp.Net Core, середовище програмування Visual Studio, PostgeSql Raspberry Pi, GPS-модуль, камера.

4. Зміст пояснювальної записки

Вступ, опис предметної області, аналіз існуючих рішень, аналіз вимог до програмного забезпечення, вибір технологій розробки та компонентів системи, розробка системи.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо)

Діаграма структури бази даних, схема архітектури сервера, структурна схема, схема взаємозв'язків, діаграма варіантів використання.

6. Дата видачі завдання 03.03.2020

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Аналіз існуючих рішень	03.03.2020 – 14.03.2020	
2	Вибір окремих компонентів пристрою	13.03.2020 – 21.03.2020	
3	Вибір засобів для програмної реалізації, розробка алгоритмів роботи програми	21.03.2020 – 28.03.2020	
4	Розробка структури програмного забезпечення та структури бази даних	28.03.2020 – 12.04.2020	
5	Написання програмних модулів	12.04.2020 – 09.05.2020	
6	Тестування та відлагодження	09.05.2020 – 13.05.2020	
7	Опис роботи	13.05.2020 – 17.05.2020	
8	Оформлення текстової документації	17.05.2020 – 01.06.2020	

Студент Антон КОМАР

Керівник Олег СОКУЛЬСЬКИЙ

АНОТАЦІЯ

Комар А.А. Автоматична система моніторингу та аналізу кількості пасажирів у транспортних засобах. КПІ ім. Ігоря Сікорського, Київ, 2020.

Проєкт містить 62 с. тексту, 29 рисунків, 5 таблиць, посилання на 18 літературних джерел, 2 додатки та 4 конструкторських документи.

Ключові слова: система моніторингу, Computer Vision, сервер, Python, C#, OpenCV, Asp.Net Core, Rasperry Pi.

Об’єктом розробки є автоматична система обробки даних про кількість пасажирів в транспортних засобах.

Метою даної розробки є впровадження в транспортних засобах розробленої автоматичної системи для моніторингу та аналізу пасажиропотоку за допомогою камер, з подальшим збереженням отриманих даних на сервері. Система допоможе контролювати завантаженість транспортних засобів на різних маршрутах, що набагато підвищить зручність пересування для пасажирів.

У дипломному проекті проведено ретельний аналіз, вибір потрібних модулей для пристрою та вибір технологій для розробки системи. Розроблено систему обробки даних про кількість пасажирів з використанням фреймворку Asp.Net Core і мови програмування C# для серверної частини, та мови програмування Python і бібліотеки OpenCV для пристрою на основі Raspberry Pi.

Отримані результати можуть бути корисними при створенні аналогічних чи подібних систем.

SUMMARY

Komar A.A. Automatic system for monitoring and analyzing the number of passengers in vehicles. Igor Sikorsky KPI, Kyiv, 2020.

The project contains 62 pages. text, 29 figures, 5 tables, links to 18 literary sources, 2 annexes and 4 design documents.

Keywords: monitoring system, Computer Vision, server, Python, C#, OpenCV, Asp.Net Core, Rasperry Pi.

The object of development is an automatic system for processing of data on the number of passengers in vehicles.

The purpose of the development is to implement in vehicles the developed automatic system for monitoring and analysis of passenger traffic with the help of cameras, with subsequent storage of the received data on the server. The system will help to control the load of vehicles on different routes, which will greatly increase the ease of movement for passengers.

The diploma project conducted a thorough analysis, selection of the necessary device modules, and selection of technologies for system development. A system for processing data on the number of passengers has been developed by using the Asp.Net Core framework and the C# programming language for the server part, and the Python programming language and the OpenCV library for the Raspberry Pi-based device.

The results obtained can be useful in creating analogical or similar systems.

Номер	Формат	Позначення	Найменування	Кільк.	Номер	Приміт
1			Документація загальна			
2						
			Знову розроблена			
	А4	ІА61.120БАК.005 ПЗ	Пояснювальна записка	62		
	А3	ІА61.120БАК.005 Д1	Автоматична система	1		
			моніторингу та аналізу			
			кількості пасажирів в			
			транспортних засобах.			
			Архітектура взаємозв’язків			
			між проектами сервера.			
	А3	ІА61.120БАК.005 Д2	Автоматична система	1		
			моніторингу та аналізу			
			кількості пасажирів в			
			транспортних засобах.			
			Схема взаємозв’язків.			
	А3	ІА61.120БАК.005 Д3	Автоматична система	1		
			моніторингу та аналізу			
			кількості пасажирів в			
			транспортних засобах.			
			Структурна схема.			
	А3	ІА61.120БАК.005 Д4	Автоматична система	1		
			моніторингу та аналізу			
			кількості пасажирів в			
			транспортних засобах.			
			Діаграма варіантів			
			використання.			

					ІА61.120БАК.005 ТП
Зм.		№ докум.	Під	Д	
Розроб.		Комар А.А.			Автоматична система моніторингу та аналізу кількості пасажирів у транспортних засобах Відомість технічного проекту
		Сокульський			
Реценз.					
Н. Контр.					
Затв.					

Літ	Ар	Аркуш
	1	1

«КПІ ім. Ігоря Сікорського»
ФІОТ, група ІА-61

Пояснювальна записка
до дипломного проєкту
на тему:
«Автоматична система моніторингу та аналізу кількості пасажирів у транспортних засобах»

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних;

ІЧ – інфрачервоний;

GPS – система глобального позиціонування (англ. – Global Positioning System);

API – прикладний програмний інтерфейс (англ. – application programming interface);

UART – універсальний асинхронний приймач-передавач (англ. – universal asynchronous receiver/transmitter);

DDD – предметно-орієнтоване програмування (англ. – Domain Driven Design);

CMOS – комплементарна структура метал-оксид-напівпровідник (англ. – complementary metal-oxide-semiconductor);

IoT – інтернет речей (англ. – Internet of Things);

AI – штучний інтелект (англ. – Artificial Intelligence);

ТЗ – транспортний засіб;

ПЗ – програмне забезпечення;

RPI – Raspberry Pi;

GPIO – інтерфейс введення / виведення (general purpose input / output);

IoC – інверсія управління (англ. – Inversion of Control);

DAL – рівень доступу до даних (англ. – Data Access Layer);

HTTP – протокол передачі гіпертексту (англ. – HyperText Transfer Protocol);

IIS – англ. – Internet Information Services;

SSD – англ. – Single Shot Detection;

DTO – об’єкт передачі даних (англ. – Data Transfer Object);

FPS – кадрів на секунду (англ. – Frames Per Second).

ВСТУП

Кожний з нас час від часу користується послугами пасажирських автомобільних перевезень, так як це один з оптимальних способів пересуватися у великих містах або при відсутності власного транспорту. В часи пік часто зустрічаються випадки перевищення встановленої кількості пасажирів для конкретного транспортного засобу. Для всіх осіб які пересуваються саме в такі періоди доби – це створює великі незручності та викликає роздратування і небажання в наступний раз користуватися таким видом транспорту.

На сьогодні існує багато різновидів систем для контролю кількості осіб в різних закритих чи відкритих приміщеннях. Класифікуються вони зазвичай за видами датчиків, які використовуються в основі таких систем. Серед них є і системи в основі яких лежить відеоспостереження. Тобто за допомогою низки вже існуючих моделей та алгоритмів «комп'ютерного бачення» проводиться обробка і аналіз отриманих з камер зображень в режимі реального часу, таким чином розпізнаються та рахуються люди які потрапили в кадр. На даний час це є найбільш точний вид ведення підрахунку людей, який також в порівнянні з іншими видами систем викликає менше труднощів, пов'язаних з технічним обслуговуванням. Та на жаль, поки що такі технології мало використовуються саме в сфері пасажирського автомобільного сполучення. Таким чином на сьогоднішній день це досі є актуальним питанням.

Тому основною метою даної дипломної роботи є проектування та створення загальної автоматичної системи моніторингу та аналізу кількості пасажирів в основі якої лежить ведення відеоспостереження всередині таких транспортних засобів.

Використання такої системи підвищить зручність пересування, що є дуже важливо. Також це надасть дані управляючим установам, використовуючи які, в залежності від часу, пори року, місцевим особливостям та ін., буде можливість оптімізувати перевезення. Тобто, наприклад, або ж збільшити або ж зменшити

кількість рейсів, автомобілів на певних маршрутах, що забезпечить економію та підвищення прибутку в даній сфері.

Для досягнення поставленої мети були вирішені наступні завдання:

– огляд існуючих рішень автоматичних систем обробки даних про кількість пасажирів в транспортних засобах;

– вибір модулів для пристрою;

– формування вимог до розроблюваного програмного забезпечення пристрою та серверу для зберігання отриманих даних;

– розробка архітектури серверу;

– проектування та створення структури бази даних;

– вибір існуючої моделі та алгоритму для найбільш оптимального обліку пасажирів.

Практичне значення даного проекту полягає у створенні автоматичної системи підрахунку пасажирів, за допомогою якої можна було б отримати доступ до даних по завантаженість транспортних засобів по окремим маршрутам.

Бакалаврський проект включає в себе наступні розділи: вступ, основні розділи, висновки, список використаних джерел із 18 найменувань та 2 додатків. Графічна частина складається з 4 креслеників формату А3. Загальний обсяг – 62 сторінки.

1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Обґрунтування доцільності розробки

Системи спостереження і ведення підрахунку людей представляють великий інтерес як і для комерційних так і для некомерційних сценаріїв. Такі системи часто використовуються для обліку людей, що входять і виходять з магазинів, також для визначення зайнятості офісних будівель. Така інформація є корисною для торгівельних установ, співробітників служби безпеки. І крім того ці системи є корисними для ведення обліку пасажирів в транспортних засобах, що допоможе операторам поїздів, водіям маршрутних таксі краще контролювати ситуацію по завантаженості. Також це допоможе оптимізувати вже існуючі маршрути, рейси та підвищити зручність користування такими транспортними засобами.

Зазвичай для реалізації такої системи, для збору інформаціі за допомогою відеоспостереження та визначення геолокації, використовують мову програмування Python і бібліотеку OpenCV, що забезпечує все потрібне для побудови програмного забезпечення з використанням «комп'ютерного бачення». Крім того, перевага саме цих технологій визначається багатьма пунктами, а саме: простотою освоєння, кількістю фахівців які добре в них розбираються та високою швидкістю розробки. Навколо цієї мови і бібліотеки вибудувалась величезна спільнота спеціалістів, які при необхідності можуть відповісти на питання чи надати рішення до проблеми яка виникла в процесі розробки.

1.1.1 Опис та аналіз предметного середовища

Автоматична система підрахунку кількості пасажирів – це програмне забезпечення, головною метою якого є збір інформації з датчиків, модулів та відео-камер та збереження цих даних на сервері для можливості доступу до них клієнтів.

Такі системи будуються на основі архітектури схожої до клієнт-серверної, але до неї ще додається третій – Шлюзний рівень [1]. Клієнтом в такій архітектурі є рівень

пристроїв, а сервером – хмарний рівень, і в загальному представляється в такому порядку – рівень пристроїв – шлюзний рівень – хмарний рівень (рисунок 1.1).

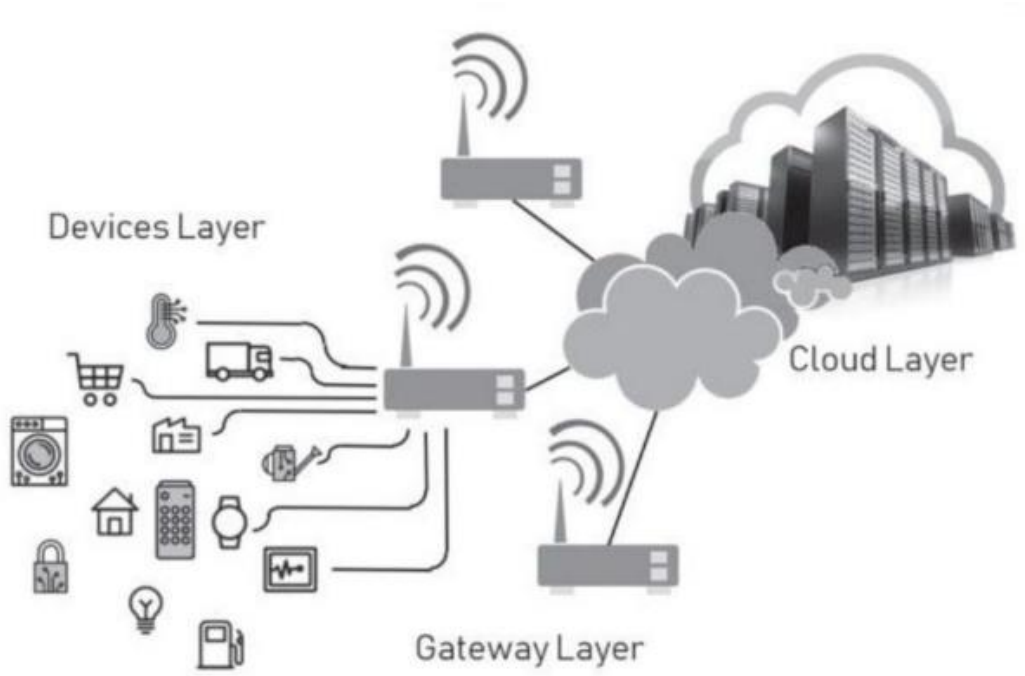

Рисунок 1.1 – Архітектура системи пристрій – шлюз – хмара [1]

Хмара представляє собою або ж приватний центр даних, або ж публічний центр який пропонується такими сервісами як Amazon, Google Cloud Platform, Microsoft Azure, чи їх комбінаціями. З моделі на рисунку 1.1 хмарний рівень являється back-end сервісом, за допомогою якого можна налагоджувати, керувати, оперувати та виділяти практичну цінність такої системи. На хмарному рівні, складні бізнес-аналітичні двигуни, штучний інтелекті (AI), веб-програми використовуються для обробки та аналізу даних, зібраних системою, і передають оброблювані дані для представлення складних моделей системи операторам та користувачам.

Рівень шлюзу є проміжним шаром в даній архітектурі і стосується обробки даних, зв’язку та управління. Шлюзи - це вузли зв'язку та обробки, які безпосередньо обслуговують рівень пристроїв.

Рівень пристроїв системи складається з вузлів, які взаємодіють з навколишнім середовищем збираючи дані. Вузли, які збирають дані, називаються датчиками. Шар пристроїв іноді називають шаром сприйняття, оскільки він сприймає або бачить навколишнє середовище за допомогою датчиків.

1.1.2 Аналіз функціональних особливостей та принцип роботи системи

Для того щоб рахувати пасажирів в транспортних засобах за допомогою відеоспостереження є два варіанти розташування камер.

В першому випадку камера встановлюється біля водія на максимальній висоті ТЗ для того, щоб за один раз порахувати усіх пасажирів які там знаходяться. Для цього висота ТЗ повинна бути не нижче 220 см, щоб камера мала змогу зафіксувати в кадрі всіх пасажирів. Але недолік такого способу в тому, що в будь-якому випадку камера не може зафіксувати деяких пасажирів із-за завантаженості ТЗ або ж якщо одна особа закриває іншу. Тобто точність підрахунків при такому положенні камери буде дуже низькою, та й не усі ТЗ мають потрібну висоту для встановлення камери таким чином.

В другому випадку камера встановлюється на усіх дверних отворах що ведуть до ТЗ, в ПЗ при обробці зображення встановлюється лінія, перетинаючи яку – ПЗ визначає в якому направленні рухається людина, відкидуючи випадки коли людина перетинає лінію і повертається назад, таким чином встановлюється кількість осіб які зайшли та вийшли з ТЗ. Знаючи що спочатку ТЗ був пустий – ми можемо встановити кількість осіб, які знаходяться всередині ТЗ на даний час. Точність цього способу буде значно вище, тому очевидно, що обирається саме цей варіант, а не перший.

Для того, щоб підвищити точність підрахунків при недостатній видимості – обираються камери з ІЧ-фільтром. Кадри зафіксовані з ІЧ-фільтром мають кращу передачу зображень при недостатній видимості, хоча і майже чорно-білі.

Встановлені камери підключаються до спільного центру обробки, яким в нашому випадку є Raspberry Pi. В основі ПЗ для обробки зображення використовується

OpenCV. RPI має підключений GPS-модуль через UART для отримання даних про точне місцезнаходження ТЗ та фіксації часу виконання збору даних. Також на кожному дверному отворі до RPI через GPIO підключається проводний магнітний контактний перемикач або ж іншими словами - дверний датчик. Він визначає коли відкриваються та закриваються двері. На основі отриманих від нього сигналів в RPI відбувається збір даних при відкритті дверей та через деякий час після закриття дверей – закінчується.

Для встановлення зв'язку з сервером є два варіанти:

– встановлення GRPS/GSM модуля, та використання сотового зв’язку, що потребує додаткових витрат на реєстрацію в мережі та трафік;

– використання наявної мережі Wi-Fi.

Так як на сьогоднішній день є тенденція з якою в кожному ТЗ встановлюють Wi-Fi зв’язок, то для встановлення зв’язку з сервером було обрано мережу Wi-Fi.

Після збору – дані отримані з GPS-модуля розкодовуються, та записуються в json-модель разом з кількістю наявних пасажирів в ТЗ на даний час та унікальним номером даного ТЗ. Модель за допомогою мережі Wi-Fi на рівні HTTPS протоколу відправляються на REST API сервера. В разі відсутності зв'язку з інтернетом через мережу Wi-Fi – дані зберігаються в локальній базі даних SQLite та повторно відправляються при встановленні з’єднання з інтернетом.

Отримані та оброблені дані на сервері зберігаються далі в базі даних PosgreSQL. В результаті ми можемо відслідкувати пасажиропотік по усім маршрутам які використовують таку систему. Також API серверу надає можливість отримання даних про наявні ТЗ, які використовують таку систему, та інформацію про зупинки по маршруту кожного такого ТЗ.

1.2 Цілі та задачі розробки

Цілі розробки даної системи можна умовно розділити на три категорії: для водіїв ТЗ, його пасажирів та для регулювальних установ.

Головна ціль створення системи для водіїв ТЗ – це полегшення контролю за кількістю пасажирів які увійшли чи вийшли відповідно.

Головна ціль створення системи для пасажирів – можливість переглядати завантаженість та місцезнаходження ТЗ потрібних маршрутів real-time або ж на основі історії за обраний час і день на попередній неділі.

1.3 Висновок до розділу

На основі проведеного аналізу технологій та функціональних можливостей для даної системи, з поміж широкого спектру було обрано окремі технології, що найбільш оптимальні для використання в наведених випадках. Встановлено порядок виконання дій та принцип функціонування даної системи. Обрано архітектурну модель побудови системи. Наведено цілі та задачі, які будуть досягнуті та вирішені відповідно, після реалізації системи по наведеному вище принципу.

Опираючись на проведений аналіз було підтверджено те, що потреба в розробці описаної автоматичної системи ведення обліку кількості пасажирів – є. Та можна сміливо стверджувати, що тема даної дипломної роботи є актуальною на сьогоднішній день.

2 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Перед тим як створити власне рішення, необхідно знайти, ретельно дослідити та вивчити вже існуючі пристрої та системи схожі по специфіці роботи до тих які плануємо розробити, також потрібно провести аналіз та оцінити позитивні та негативні аспекти кожного з них.

2.1 Класифікація систем для обліку пасажирів

В залежності від того який різновид датчиків використовується в основі пристрою, можна класифікувати системи за використанням датчиків на такі:

а) інфрачервоні датчики;

Датчики активного типу, що складаються з передавача і приймача; вони створюють точковий промінь і працюють як виявлення вкл/викл.

Інфрачервоні випромінювачі зазвичай встановлюються паралельно один одному, так що переривання світлодіода відбувається в напрямку перетину і, таким чином, напрямок входу можна відрізнити від вихідного: через зміну «смуг» Тип інфрачервоного датчика також визначається як «бар'єрні датчики». Недолік системи з таким датчиком в тому, що він не здатний обробити групу людей, яка проходить через нього за раз, так як це потрібно.

б) датчик на килимі;

Датчик на килимі - розміщення на сходинках автобуса, трамвая або поїзда – реєструє пасажирів, коли вони наступають на килимок. Це рішення розроблено кількома компаніями, що працюють в транспортній галузі: в системі підрахунку використовуються ножні мати розташовані в безпосередній близькості до воріт автомобіля. Недолік пристрою з таким датчиком в тому, що він потребує частого додаткового технічного обслуговування, крім того в проміжок між порогом та датчиком може попадати велика кількість бруду та сміття. Також як і в попередньому

випадку – системи з таким датчиком не здатні розрізняти кількість осіб в групі людей які проходять через нього за один раз. Тобто в такому випадку для ведення правильного обліку потрібно щоб пасажири проходили до ТЗ по одному.

в) оптичні датчики і відеоспостереження.

Оптичні датчики допускають ту ж точність, що і датчики на килимах, але з меншими труднощами, пов'язаними з технічним обслуговуванням (немає рухомих частин і, отже, ризик проникнення бруду нижче) і установки (в разі тільки одного датчика, як в пасивних інфрачервоних датчиках).

Що стосується систем відеоспостереження, на ринку є різні рішення, але майже всі вони засновані на двох стереоскопічних камерах, здатних захоплювати зображення в області під камерами. Ці системи, встановлюються над дверима, здатні з високою точністю ввести облік кількості пасажирів, що входять або виходять з автобусів, трамваїв або поїздів. Ці системи зазвичай мають вбудовані світлодіоди, які дозволяють системі працювати практично в будь-яких умовах освітлення. Крім того, регульований поріг підрахунку дозволяє вимірювати висоту пасажирів, щоб розрізняти дітей та дорослих.

2.2 Приклади існуючих алгоритмів підрахунку людей за допомогою відеоспостереження

Пошук та відстеження людей у громадських та / або закритих просторах є об'єктом широких досліджень. Умови, в яких відбувається відстеження, є вирішальним значенням для успіху. Використовуючи вид камери зверху – Терада та ін. [2] створили систему, яка може визначати рух людей в конкретному напрямі і таким чином рахувати людей під час перетину віртуальної лінії (рис. 2.1). Вид зверху вниз дозволяє уникнути проблеми оклюзії, коли групи людей проходять через поле зору камери. Для визначення напрямку людей використовується образ простору-часу.

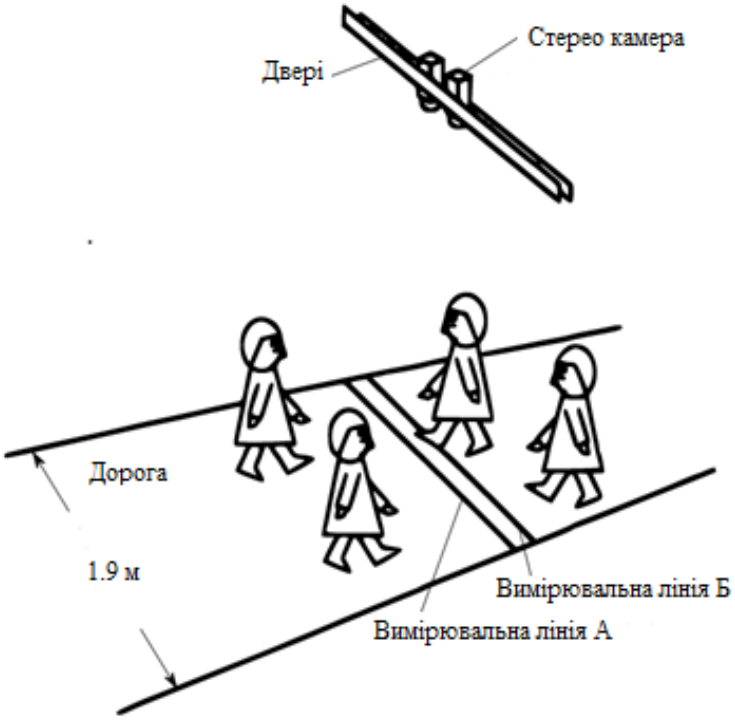

Рисунок 2.1 – Ілюстрація системи стереокамери Теради

Беймер також використовує стерео-бачення для відстеження людей [3]. Відстеження на основі шаблонів дозволяє відмовитись від відстеження людей, коли вони зникають с поля зору, усуваючи помилкові позитиви у відстеженні людей (рисунок 2.2).

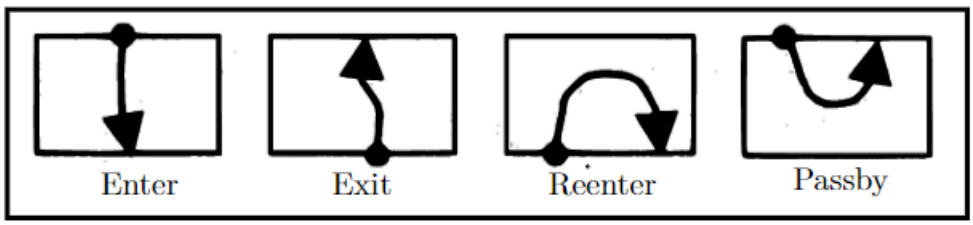

Рисунок 2.2 – Категорії Беймера для відстеження [3]

Хашимото та ін. використовували спеціалізовану систему візуалізації, розроблену власноруч (використовуючи чутливу кераміку, механічні деталі та ІЧ-прозорі лінзи) [4]. Вони розробили систему на основі масиву, яка може підраховувати людей, коли вони проходять через 2-метрові двері з точністю 95%. Для роботи в хороших умовах

система вимагає відстані не менше 10 см між людьми, що проходять, щоб їх розрізнити і таким чином рахувати їх двома окремими особами.

Тесей та ін. використовувати сегментацію зображення для відстеження людей та обробки оклюзій [5]. Для вилучення таких регіонів їх метод використовує віднімання фону. Вони використовують площу блоку, висоту та ширину, обмежуючи площу блоку, периметр та середній рівень сірого для відстеження блоків. Запам'ятовуючи всі ці особливості з часом, алгоритм може вирішити проблему злиття та відокремлення блоків, що відбувається із-за оклюзії. Коли блоки зливаються під час оклюзії, створюється новий блок з іншими ознаками. Однак цей новий блок зберігає дані з функцій блоків, які його формують. Отже, коли блоки відокремлюються назад, алгоритм може присвоїти оригінальні мітки оригінальним блокам.

Робота Шио [6] зосереджена на виявленні оклюзій людей в алгоритмі сегментації заднього плану шляхом імітації перцептивного групування людини. По-перше, алгоритм аналізує рух за допомогою різних кадрів і використовує ці дані, щоб допомогти алгоритму віднімання фону визначити межі між окклюдованими особами. Цей метод використовує імовірнісну модель об'єкта, яка містить інформацію про ширину, висоту, напрямок та крок злиття / розщеплення, як показано в [5]. Було встановлено, що використання об'єктної моделі є хорошим вдосконаленням сегментації та можливим способом вирішення проблеми оклюзій. Але використання перцептивного групування є абсолютно неефективним у деяких ситуаціях, як, наприклад, група людей, що рухається в одному напрямку з майже однаковою швидкістю.

Як простий та ефективний підхід, Секстон та ін. використовували спрощений алгоритм сегментації [7]. Вони протестували свою систему на паризькому залізничному вокзалі і отримали помилку в межах від 1% до 20%. Їх система використовує простий метод віднімання фону, щоб ізолювати людей від фону, беручи лише центроїди блоків, щоб здійснити відповідність між кадрами. Для покращення надійності системи фонова модель постійно оновлювалася, зменшуючи вплив

зовнішнього середовища на систему. Камеру повісили у верхньому положенні, що також зменшило оклюзії та спростило проблему виявлення блоків.

В [8] Сеген концентрується на обробці зображень після сегментації. Вони витягують блоки за допомогою простого методу віднімання фону, а потім відстежують їх ознаки між кадрами. Шлях кожного блоку зберігається та використовуються для виявлення перетину та напрямку при перетині віртуальної лінії. Ця система не справляється з проблемами оклюзії, тому її продуктивність значно зменшується в переповнених умовах.

У [9] Гері Конрад також використовує накладну камеру для спрощення проблеми оклюзій. Щоб подолати проблеми зміни освітленості, вони використовують різницю у послідовних кадрах, а не фонове віднімання. Через обмежену обчислювальну потужність їх алгоритм був розроблений лише для того, щоб діяти у невеликій прямокутній області зображення. Вони змогли досягти 95,6% рівня точності з-понад 7491 чоловік.

У роботі [10] Джу-Шенг пропонує нову 3D-модель підсвічування конусоподібної форми та поєднує довгострокову кольорову модель фону та короткочасну кольорову фонову модель для вирішення проблем у [9].

У роботі [11] Харітаоглу та Флікнер застосовують інший метод вирішення проблеми відстеження людей у реальному часі. Щоб сегментувати передній план, вони використовують віднімання фону на основі кольору та інтенсивності пікселів. Пікселі класифікуються на три різні групи: передній план, фон та тінь. Регіони переднього плану сегментуються на окремих людей, використовуючи два обмеження руху: по часу та глобально. Для того, щоб відстежувати осіб, алгоритм використовує модель зовнішнього вигляду, засновану на кольорі, щільності границь та відстеженні зсуву.

У роботі [12] нам представлена система збереження конфіденційності, яка оцінює розмір неоднорідних натовпів, що складається з пішоходів, які рухаються в різних напрямках, не використовуючи явної сегментації об'єктів або відстеження. По-перше,

натовп сегментується на компоненти однорідного руху, використовуючи суміш динамічної моделі текстур руху [13]. По-друге, набір простих ознак видобувається з кожної сегментованої області, а відповідність між ознаками та кількістю людей на сегмент вивчається за допомогою регресії Гауссового процесу.

У [14] гістограма квантованих локальних описів ознак використовувалась для представлення та співставлення відстежених об'єктів. Цей метод виявився ефективним для співставлення об'єктів та класифікації у програмах пошуку зображень, де можуть бути вилучені описи ознак. Ця система підходить для роботи в режимі реального часу.

2.3 Приклади існуючих пристроїв підрахунку людей на основі відеоспостереження

а) AXIS People Counter;

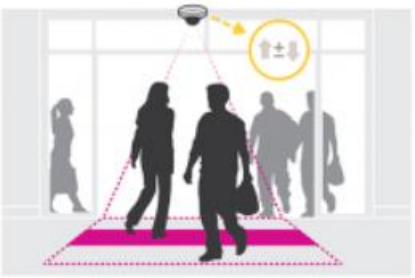

Рисунок 2.3 – AXIS People Counter

Лічильник людей AXIS може використовуватися для покриття одного або двох дверей, оскільки кілька камер із вбудованим програмним забезпеченням можуть бути пов'язані для забезпечення синхронізованого підрахунку (рисунок 2.3). Віддалене налаштування, управління та моніторинг дозволяють переглядати статистику з кількох камер та локацій одночасно.

Дані зберігаються до 90 днів і доступні до перегляду двома способами:

– за допомогою стандартного веб-браузера для доступу до веб-інтерфейсу камери;

– завантажуючи статичні вихідні дані, доступні в різних форматах, через відкритий API в камеру.

б) Brickstream 2D.

Рисунок 2.4 – Brickstream 2D

Ключові можливості:

Brickstream 2D ідеально підходить для проектів підрахунку кількості людей (рисунок 2.4). Він надає дані по кількості людей, які входять та виходять із дверних отворів. Щоб дані були завжди доступними, Brickstream 2D зберігає до 10 днів ведення підрахунків, тобто дані не втрачаються, якщо мережа Ethernet недоступна.

2.2 Аналіз і порівняння

Розглянуті алгоритми ведення підрахунку людей є відображенням того, що система може мати різноманітні варіації в залежності від обраного алгоритму обробки зображення. Проаналізувавши наведені алгоритми та представлені після них приклади реалізації таких пристроїв, можна виділити основні моменти, які мають бути присутні у розробленій системі:

– найоптимальніше розміщення камери – з видом зверху вниз над входом/виходом, що допоможе запобігти виникненню проблеми оклюзії (перекриття

однієї особи іншою), рахування людей в такому випадку буде проведено при пересіченні віртуальної лінії;

– дані з камери мають тимчасово зберігатись локально на пристрої, при відсутності доступу до інтернету;

– також дані отримані з пристроїв мають бути відкриті для доступу через API на протязі якогось часу.

2.3 Висновок до розділу

Для того, щоб створити свою оптимальну автоматичну, необхідно детально вивчити вже існуючі рішення і виділити у них всі сильні та слабкі сторони. Це необхідно для того, щоб на основі цих даних розробити продукт, який би перевершив розглянуті аналоги уникне недоліків, притаманних ним.

Було розглянуто ряд алгоритмів в системах для рахування людей та описано плюси і мінуси їх використання. Також було приведено приклади вже готових систем та описано їх ключові можливості. На основі цієї інформації було проведено аналіз та встановлення функцій, що мають бути присутні в такій системі.

3 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Сценарії використання серверу системи

На кресленику ІА61.120БАК.005 Д4 наведена діаграма варіантів використання. Дивлячись на цю діаграму ми можемо виділити увесь можливий функціонал системи, який наведено в таблиці 3.1.

Таблиця 3.1 – Варіанти використання системи

Шифр варіанту використання	Назва варіанту використання
UC-1	Авторизація (Log in).
UC-1.1	Перегляд усіх транспортних засобів.
UC-1.1.1	Перегляд усіх даних про зупинки даного транспортного засобу.
UC-1.2	Редагування транспортного засобу (Edit vehicle).
UC-1.3	Додавання транспортного засобу (Add vehicle).
UC-1.4	Видалення транспортного засобу (Delete vehicle).
UC-1.5	Видалення маршруту (Delete route).
UC-1.6	Перегляд усіх маршрутів.
UC-1.7	Додавання маршруту (Add route).
UC-1.8	Редагування маршруту (Edit route).
UC-2	Вихід з системи адміністрування(Log out).

3.2 Сценарій роботи всієї системи

На кресленику ІА61.120БАК.005 Д3 наведена структурна схема автоматичної системи обробки даних про кількість пасажирів в транспортних засобах.

Система складається з двох підсистем: пристрій і застосунок.

В пристрої є такі компоненти:

а) мікрокомп'ютер Raspberry Pi;

б) камера Pi Noir;

в) GPS- NEO-6M-0-001;

г) вбудований в RPI WiFi модуль;

е) магнітний контактний перемикач (дверний датчик);

є) база даних SQLite.

Друга підсистема складається з таких компонентів:

а) клієнт;

б) сервер (Web API);

в) база Даних PostgreSQL.

На основі структурної схеми було побудовано функціональну схему системи, яка наведена на кресленику ІА61.120БАК.005 Д2.

З побудованої схеми можна виділити такі функціональні можливості:

а) сповіщення про відкриття / закриття дверей які надходять від магнітного контактного перемикача до Raspberry Pi, на основі цього сповіщення виконуються наступні пункти;

б) передача зображення від камери Pi Noir до Raspberry Pi для подальшого його обробки;

в) передача даних від GPS-модулю NEO-6M до Raspberry Pi, де з цих даних виділяється точне місцеположення пристрою;

г) мікроком’ютер RPI передає отримані дані про кількість пасажирів, місцеположення та ID ТЗ до вбудованного Wi-Fi модулю, потім у відповідь від

модулю приходить відповідь серверу, на основі якої вирішується чи потрібно зберігати дані локально в SQLite для повторного відправлення пізніше;

д) вбудований Wi-Fi модуль, в залежності від того чи є підключення до мережі інтернет, відправляє дані до серверу та у відповідь отримую від сервера повідомлення про успішність виконання запиту;

е) сервер оброблює отримані дані з пристрою та зберігає їх до бази даних PostgeSQL та у відповідь відправляє повідомлення про успішність запиту, в той же час сервер отримує та обробляє запити які надходять з клієнта та або ж відправляє клієнту потрібні дані, або ж відправляє повідомлення про успішність операції.

3.3 Висновок до розділу

У даному розділі на базі інформації, що була зібрана та проаналізована у попередніх розділах, було створено діаграму варіантів використань, а також були побудовані структурна та функціональна схеми системи, відповідно до яких було виділено компоненти системи та їх функціональні задачі. Варто зазначити, що правильно складені вимоги слугують базою для частин майбутньої системи, чим полегшують подальшу розробку та тестування програмного забезпечення.

4 ВИБІР ТЕХНОЛОГІЙ РОЗРОБКИ ТА КОМПОНЕНТІВ СИСТЕМИ

4.1 Переваги обраних компонентів системи

Було обрані такі компоненти для системи:

а) основним центром для збору та обробки даних було обрано міні-комп’ютер Raspberry Pi 3B+;

Рисунок 4.1 – Raspberry Pi 3B+

Raspberry Pi 3 Model B+ - одноплатний комп'ютер від Raspberry Pi Foundation, який працює на базі оновленого 4х-ядерного 64-бітного SoC Broadcom BCM2837B0 і збільшеною тактовою частотою 1.4GHz (рисунок 4.1). Бездротовий модуль Wi-Fi дводіапазонний стандарту IEEE 802.11ac, а Bluetooth - 4.2 BLE. Плата володіє Gigabit Ethernet, що працює через USB 2.0 шину, що забезпечить швидкість передачі даних до 300Mbps. Обсяг оперативної пам'яті - 1GB RAM.

Raspberry Pi 3 Model B+ відрізняється високим рівнем надійності, простотою налагодження, величезною спільнотою і високою якістю.

б) Raspberry Pi NoIR Camera Board - Infrared-sensitive Camera;

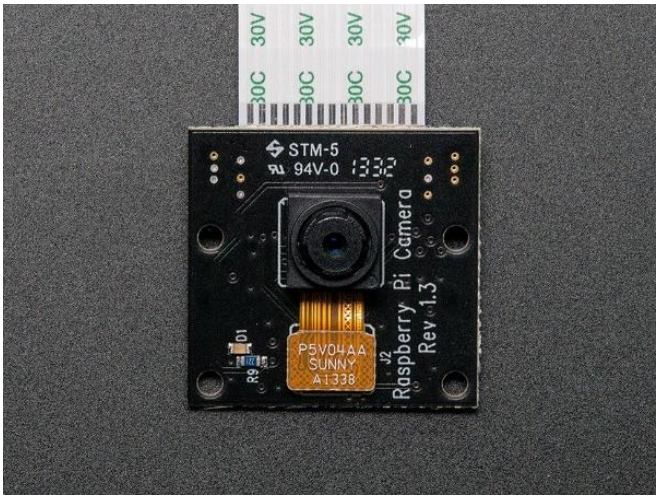

Рисунок 4.2 – Raspberry Pi NoIR Camera Board

Модуль камери Raspberry Pi NoIR - це спеціально розроблена надбудова для Raspberry Pi в якій не встановлено «ІЧ-фільтр» (рисунок 4.2). Камера кріпиться до Raspberry Pi через одне з двох маленьких гнізд на верхній поверхні плати. Цей інтерфейс використовує спеціальний інтерфейс CSI, який був розроблений спеціально для взаємодії з камерами. Шина CSI здатна працювати з надзвичайно високими швидкостями передачі даних і переносить виключно піксельні дані. Raspberry Pi не використовується.

Сам датчик має роздільну здатність в 5 мегапікселів і має вбудований об'єктив з фокусом. Що стосується нерухомих зображень, камера здатна відтворювати статичні зображення розміром 2592 x 1944 пікселів, а також підтримує відео 1080p30, 720p60 і 640x480p60 / 90.

Ця версія також добре підходить для нічного бачення. Тобто в умовах недостатньої видимості камера зможе точніше рахувати пасажирів в порівняні зі звичайною камерою з ІЧ-фільтром.

в) GPS приймач NEO-6M-0-001 з активною антеною;

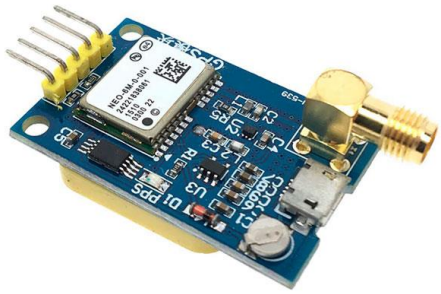

Рисунок 4.3 – GPS приймач NEO-6M-0-001 з активною антеною

GPS-приймач NEO-6M-0-001 на базі чіпа Ublox NEO-6M STM являє собою автономний GPS-пристрій з високопродуктивним процесором позиціонування u-blox 6 (рисунок 4.3). Надійний GPS модуль пропонує безліч варіантів підключення з портативним розміром.

г) магнітний контактний перемикач (дверний датчик);

Цей датчик по суті є перемикачем, укладеним у пластикову оболонку (рисунок 4.4). Зазвичай перемикач є "відкритим" (немає з'єднання між двома проводами). Інша половина - магніт. Коли магніт знаходиться на відстані більше 13 мм, перемикач закривається. Їх зазвичай використовують для виявлення, коли двері чи шухляди відкриті, тому у них є кріпильні вкладки та гвинти.

Технічні деталі:

- відкритий перемикач;
- корпус з ABS;
- номінальний струм: 100 мА максимум;
- номінальна напруга: 200 В постійного струму максимум;
- максимальна відстань закриття між пластинками: 15мм;

Рисунок 4.4 – Блок живлення для Raspberry PI

д) блок живлення для Raspberry PI - 5V, 3A з USB / MiscroUSB кабелем.

Рисунок 4.5 – Блок живлення для Raspberry PI

Пристрій живиться від стабілізованого джерела живлення з вихідною напругою 5В (рисунок 4.5). В автономному режимі пристрій живиться від акамулятора.

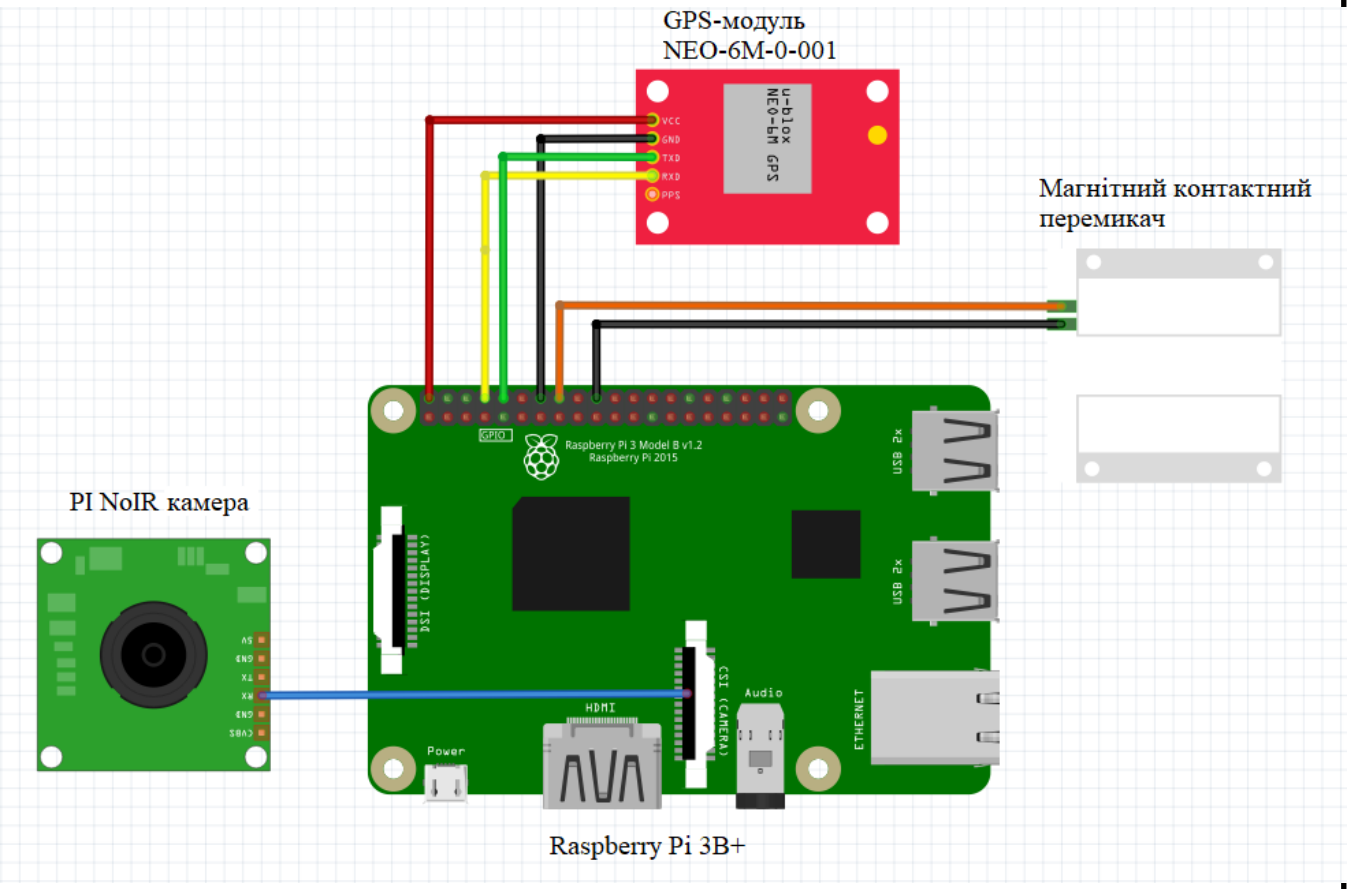

Рисунок 4.6 – Схема підключення окремих модулів до Raspberry Pi

Всі компоненти наведені вище підключаються за схемою наведеною на рисунку 4.6, а саме:

а) підключення GPS-модулю NEO-6M-0-001 до RPI по UART;

Таблиця 4.1 – Підключення GPS-модулю до RPI

GPS-модуль NEO-6M-0-001	Raspberry Pi GPIO
VCC	2 – VCC – 5V

Продовження таблиці 4.1.

GND	14 – GND
TXD	10 – GPIO 15 (TXD)
RXD	8 – GPIO 14 (RXD)

б) підключення магнітного контактного перемикача до RPI:

Таблиця 4.2 – Підключення магнітного контактного перемикача

Магнітний контактний перемикач	Raspberry Pi GPIO
1	16 – GPIO 23
2	20 - GND

в) підключення камери Pi NoIR відбувається одним шлейфом до відеовходу CSI (Camera Serial Interface).

4.2 Переваги обраних технологій розробки

При створенні програмних продуктів слід вивчити існуючі технології і практики їх використання для того, щоб обрати для розробки найкращі та найсучасніші з них.

Вибір технології залежить від багатьох факторів пов’язаних як самою системою, так і з командою розробки.

Перший фактор, який впливає на вибір технології – стек навичок команди розробки. У більшості випадків кожен розробник має технології, середовища розробки, мови програмування або розмітки, яким він надає перевагу та з якими він найчастіше працює.

Ще одним фактором при виборі технології є особливості і вимоги до системи, що розробляється. Такими факторами можуть слугувати навантаження на систему, швидкість виконання, сумісність технології з апаратною частиною тощо.

Також варто зазначити, що деякі технології можуть базуватися на інших, використовувати їх або бути сумісними з ними.

При розробці системи для пристрою були використані наступні технології:

а) мова програмування Python;

Python – об’єктно-орієнтована мова програмування високого рівня з динамічною типізацією. [15]

Особливості:

- синтаксис цієї мови дуже простий і короткий, що надає можливість швидко писати код, а також спрощує його читання;

- Python є мовою програмування високого рівня, що означає, що програмісту не потрібно знати системну архітектуру та займатися керуванням пам’яттю;

- мову програмування Python підтримують майже всі сучасні платформи, тому код, написаний на одній з них можна запустити на іншій без змін;

- Python є інтерпретованою мовою програмування, тобто вона виконується рядок за рядком, що може бути корисним для налагоджування коду;

- функціонал цієї мови можна розширити за допомогою підключення бібліотек, написаних іншою мовою (зазвичай C або C++);

- Python має велику кількість вбудованих бібліотек для різноманітних завдань; наприклад, є бібліотеки для роботи з регулярними виразами, базами даних, зображеннями, генерації документації, тестування програмного забезпечення тощо;

- Python – мова з динамічною типізацією, що означає те, що немає необхідно вказувати тип даних заздалегідь.

У даному проекті Python – це основна мова програмування на міні-комп’ютері Raspberry Pi.

б) система керування базами даних SQLite;

SQLite – це система керуванням базами даних на базі мови програмування C. SQLite є вбудованою у всі мобільні пристрої та більшість сучасних комп’ютерів. [16]

Особливості:

- SQLite не потребує ніяких налаштувань;

- уся база даних зберігається в одному файлі;

- підтримує бази даних великого розміру;

- вбудована у більшість сучасних платформ, як от: Android, *BSD, iOS, Linux, Mac, Solaris, VxWorks, and Windows (Win32, WinCE, WinRT); [17]

- має легкий у використанні інтерфейс програмування застосунків (Application Programming Interface);

- не потребує встановлення додаткових залежностей.

Система керуванням базами даних SQLite була обрана для розробки цього проекту через те, що вона є вбудованою у мову програмування Python, а також через те, що вона зберігає усі дані в одному файлі, що значно спрощує роботу з нею.

в) бібліотека OpenCV;

OpenCV (Open Source Computer Vision Library) – open-source бібліотека програмного забезпечення для комп’ютерного бачення та машинного навчання.

OpenCV був побудований для створення загальної інфраструктури для програм з використанням комп’ютерного бачення та для прискорення використання машинного сприйняття в комерційних продуктах.

Особливості:

- бібліотека має велику кількість вбудованих алгоритмів;

- розвинута спільнота користувачів;

- багато прикладів використання в інтернеті;

- в рази полегшує використання ком'ютерного бачення в будь-якому проекті.

г) бібліотека NumPy;

NumPy - це розширення мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами. Проект з відкритим кодом, який має на меті полегшити використання чисельних обчисленнь з Python. Він був створений у 2005 році, спираючись на ранню роботу бібліотек «Numerical» та «Numarray».

д) бібліотека Dlib;

Dlib – це сучасний інструментарій на C++, що містить алгоритми машинного навчання і інструменти для створення складного програмного забезпечення для вирішення завдань.

Особливості:

- документація;
- високоякісний портабельний код;
- велика кількість доступних алгоритмів машинного навчання;
- низка доступних чисельних алгоритмів та методів;
- обробка зображень.

е) бібліотека Imutils;

Бібліотека Imutils включає серію зручних функцій, щоб зробити основні функції обробки зображень, такі як переклад, обертання, зміна розміру, скелетонізація, показ зображень Matplotlib, сортування контурів, виявлення країв та багато іншого простіше з OpenCV і з Python 2.7 та Python 3.

є) бібліотека Pynmea2;

Pynmea2 – це бібліотека для мови Python, яка виділяє дані з простоколу NMEA 0183, який використовується в GPS приймачах для передачі отриманих даних.

ж) бібліотека PySerial;

Цей модуль інкапсулює доступ до послідовного порту. Він пропонує додаткові реалізації для Python, що працюють у Windows, OSX, Linux, BSD (можливо, будь-яка система, сумісна з POSIX) та IronPython. Модуль під назвою "serial" автоматично вибирає відповідну реалізацію.

з) бібліотека Urllib;

Urllib - це пакет, який включає декілька модулів для роботи з URL-адресами:

- urllib.request для відкриття та читання URL-адрес;

- urllib.error, що містить винятки, викликані urllib.request;

- urllib.parse для розбору URL-адрес;

- urllib.robotparser для розбору файлів robots.txt.

и) бібліотека RPi.GPIO.

Raspberry-gpio-python або RPi.GPIO - це модуль Python для управління інтерфейсом GPIO на Raspberry Pi. Модуль RPi.GPIO встановлений за замовчуванням на останніх версіях Raspbian Linux. RPi.GPIO підтримує посилання на піни GPIO, використовуючи або фізичні номери пінів на роз'ємі GPIO, або використовуючи імена каналів BCM від Broadcom SOC, до яких підключені піни. Наприклад, контакт 24 - це канал BCM GPIO8.

При розробці проєкту для серверу були використані наступні технології:

а) мова програмування C#;

C # - сучасна об'єктно-орієнтована і типобезпечна мову програмування. C# відноситься до широко відомого сімейства мов C, і здається добре знайомим кожному, хто працював з C, C ++, Java або JavaScript.

C # є об'єктно-орієнтованою мовою, але підтримує також і компонентно-орієнтоване програмування. Розробка сучасних додатків все більше тяжіє до створення програмних компонентів у формі автономних і самоописних пакетів, що реалізують окремі функціональні можливості. Головна особливість таких компонентів в тому, що вони являють собою модель програмування з властивостями, методами і подіями. У них є атрибути, що надають декларативні відомості про

компонент. Вони включають в себе власну документацію. C # надає мовні конструкції, які безпосередньо підтримують таку концепцію роботи. Завдяки цьому C# підходить для створення і застосування програмних компонентів.

В проекті мова C# – це основна мова програмування на сервері.

б) платформа Asp.Net Core;

Платформа ASP.NET Core представляє технологію від компанії Microsoft, призначену для створення різного роду веб-додатків: від невеликих веб-сайтів до великих веб-порталів і веб-сервісів.

Розробка над платформою почалася ще в 2014 році. У червні 2016 року вийшов перший реліз платформи. А в грудні 2019 року ми побачили версію ASP.NET Core 3.1, яка власне і буде використана в поточному проекті.

ASP.NET Core може працювати поверх крос-платформного середовища .NET Core, яка може бути розгорнуте на основних популярних операційних системах: Windows, Mac OS, Linux. А для розгортання веб-додатки можна використовувати традиційний IIS, або крос-платформний веб-сервер Kestrel.

в) Swagger;

Swagger - це фреймворк для специфікації RESTful API. Його перевага полягає в тому, що він дає можливість не тільки інтерактивно переглядати специфікацію, а й відправляти запити - так званий Swagger UI (рисунок 4.5).

Тобто Swagger дуже корисний для тестування розроблених backend сервісів, коли іншої змоги це зробити просто немає. Також це віздразу готова документація по доступним ендпоінтам API сервера. Дуже корисно, наприклад, для застосунку, який повинен взаємодіяти із зовнішньою системою, а домовитися один з одним в текстовому форматі мало. Інтерактивність проявляється в тому що з документації можна робити HTTP-запити, а сама документація залежить від коментарів у програмному коді.

Після написання специфікації OpenAPI та засоби Swagger можуть сприяти подальшій розробці API різними способами: Swagger Codegen для генерації клієнтських бібліотек для API більш ніж 40 мовами, інтерфейс Swagger для створення інтерактивної документації API, яка дозволяє користувачам випробовувати виклики API безпосередньо у веб-переглядачі, специфікацію для підключення інструментів, пов'язаних з API.

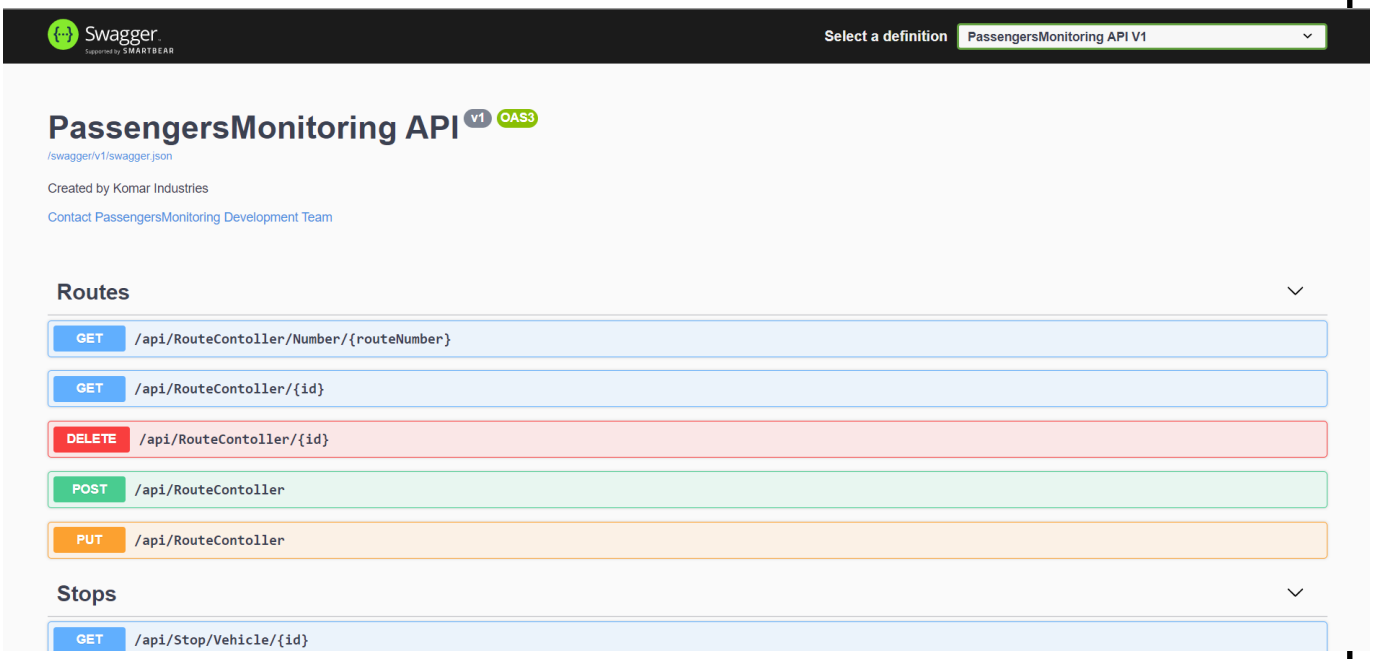

Рисунок 4.7 – Swagger UI

г) AutoMapper;

AutoMapper - це проста маленька бібліотека, створена для вирішення оманливо складної проблеми - позбавлення від коду, який відображає один об’єкт на інший.

д) FluentValidation;

FluentValidation - це бібліотека .NET для побудови строго типізованих правил перевірки. Це використовується для того, щоб в систему не потрапили дані які порушують логіку нашої системи. А дана бібліотека істотно полегшує написання таких правил.

е) Entity Framework Core;

Entity Framework Core (EF Core) являє собою об'єктно-орієнтовану, легковажну технологію від компанії Microsoft для доступу до даних. EF Core є ORM-інструментом (object-relational mapping - відображення даних на реальні об'єкти). Тобто EF Core дозволяє працювати з базами даних, але є більш високим рівнем абстракції: EF Core дозволяє абстрагуватися від самої бази даних і її таблиць і працювати з даними незалежно від типу сховища. Якщо на фізичному рівні ми

оперуємо таблицями, індексами, первинними і зовнішніми ключами, то на концептуальному рівні, який нам пропонує Entity Framework, ми вже працюємо з об'єктами.

Entity Framework Core підтримує безліч різних систем баз даних. Таким чином, ми можемо через EF Core працювати з будь-якої СУБД, якщо для неї є потрібний провайдер.

За замовчуванням на даний момент Microsoft надає ряд вбудованих провайдерів: для роботи з MS SQL Server, для SQLite, для PostgreSQL.

є) база даних PostgreSQL;

PostgreSQL - це потужна об'єктно-реляційна база даних із відкритим кодом, яка використовує та розширює мову SQL у поєднанні з багатьма функціями, які безпечно зберігають та масштабують найскладніші навантаження даних.

PostgreSQL заслужив потужну репутацію за свою перевірену архітектуру, надійність, цілісність даних, надійний набір функцій, розширюваність та відданість спільноті з відкритим кодом, що стоїть за цим програмним забезпеченням, щоб послідовно пропонувати ефективні та інноваційні рішення. PostgreSQL працює на всіх основних операційних системах, сумісний з ACID з і має потужні додатки, такі як популярний розширювач геопросторових баз даних PostGIS. Не дивно, що PostgreSQL став реляційною базою даних з відкритим кодом для багатьох людей і організацій.

ж) розширення Postgis для PostgreSQL;

PostGIS - це розширення для об'єктно-реляційної бази даних PostgreSQL. PostGIS додає додаткові типи (геометрія, географія, растр та інші) до бази даних PostgreSQL. Він також додає вдосконалення функцій, операторів та індексів, які застосовуються до цих просторових типів. Ці додаткові функції, оператори, прив’язки до індексу та типи збільшують потужність основної СУБД PostgreSQL, роблячи її швидкою, багатофункціональною та надійною системою управління просторовою базою даних.

В проекті PostGIS буде використовуватись для роботи з GPS даними. Це розширення забезпечує багато готових рішень для розрахунків пов'язаних з геолокацією і також дозволяє проводити такі розрахунки прямо в базі даних.

з) розширення Uuid-ossp для PostgreSQL.

Модуль uuid-ossp забезпечує функції для генерації універсальних унікальних ідентифікаторів (UUID) за допомогою одного з декількох стандартних алгоритмів. Існують також функції для створення певних спеціальних констант UUID.

4.3 Висновок до розділу

У цьому розділі на підставі аналізу предметної області, огляду існуючих рішень, вимог до розроблюваної системи та особистого досвіду було обрано технології, які були використані при розробці даної системи. Також було описано особливості, переваги та недоліки у використанні наведених технологій.

5 РОЗРОБКА СИСТЕМИ

5.1 Структура проекту сервера

На кресленику ІА61.120БАК.005 Д1 наведено взаємозв’язки між проєктами, тобто архітектуру використану в серверній частині системи. Як можна там побачити – в проекті за основу використовується так званий підхід «Onion architecture» або ще його називають «Clean Architecture». Схему розподілення функціоналу в такій архітектурі можна побачити на рисунку 5.1.

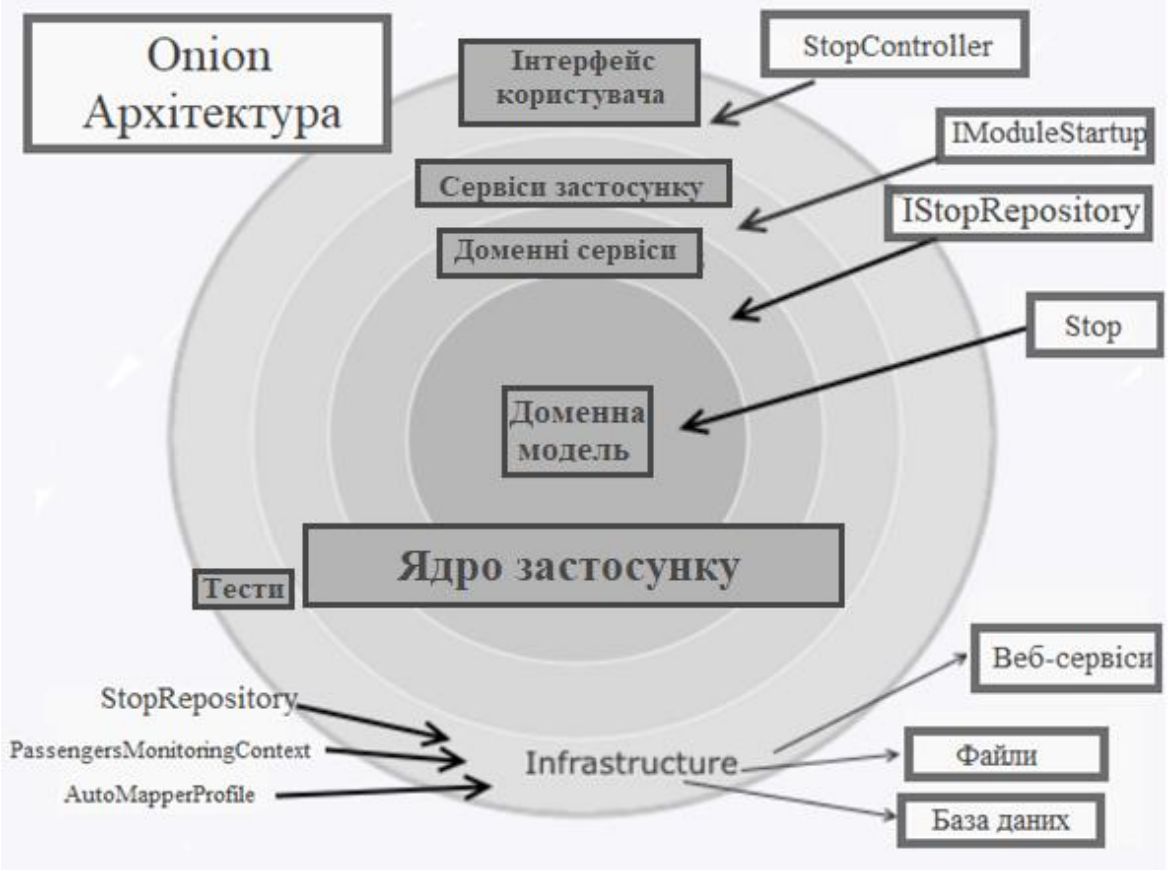

Рисунок 5.1 – «Onion architecture» використана в проєкті

Тобто це підхід який базується на використанні доменної логіки в якості ядра застосунку. На рисунку 5.2 можемо побачити список проєктів серверної частини.

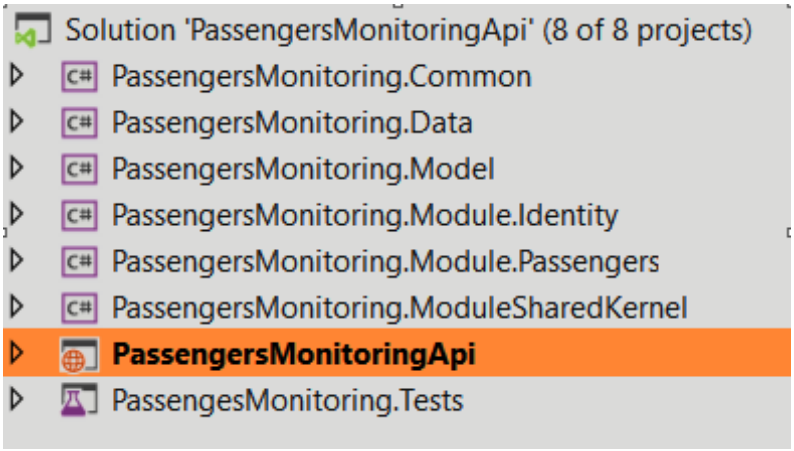

Рисунок 5.2 – Список проектів серверу

Структура проєкту, зображена на рисунку 5.2, складається з наступних компонентів:

- PassengersMonitoring.Common – являється загальним для всіх проєктів і не залежить від доменної логіки, він вирішує тільки свої локальні задачі;

- PassengersMonitoring.Model – містить усі потрібні моделі для вбудованної об’єктно-реляційної проекції (Object-relational mapping), на базі яких створюються таблиці в базі даних;

- PassengersMonitoring.Data – залежить від сутностей в проекті Model для побудови міграцій, містить DbContext (клас через за допомогою якого ми можемо проводити операції над даними які зберігаються в БД), міграції, конфігурації, що відносяться до DAL;

- PassengersMonitoring.Module.Identity – проєкт в якому реалізована логіка для авторизації та регістрації користувачів;

- PassengersMonitoring.Module.Passengers – проєкт в якому реалізована логіка для взаємодії з даними по маршрутам, зупинкам та транспортним засобам;

- PassengersMonitoring.ModuleSharedKernel – проєкт який містить загальні класи які використовуються в таких проєктах як Identity та Passengers.

-PassengersMonitoring.Tests – проєкт, в якому містяться модульні тести, що допомагають протестувати функціонал системи.

Проєкти з Module.Identity і Module.Passengers використовують IoC-контейнер для ін’єкції залежностей. Таким чином виходить досягти максимальної ізоляції коду домена від інфраструктури.

Разом з цим використовується ще й такий підхід як CQRS (command query responsibility segregation). Тобто запити на запис та читання даних з бази розділяються в окремі класи з ціллю максимально їх оптимізувати. Такий пілхід можна спостерігати в проєктах Module.Passengers та Module.Identity (рисунок 5.2).

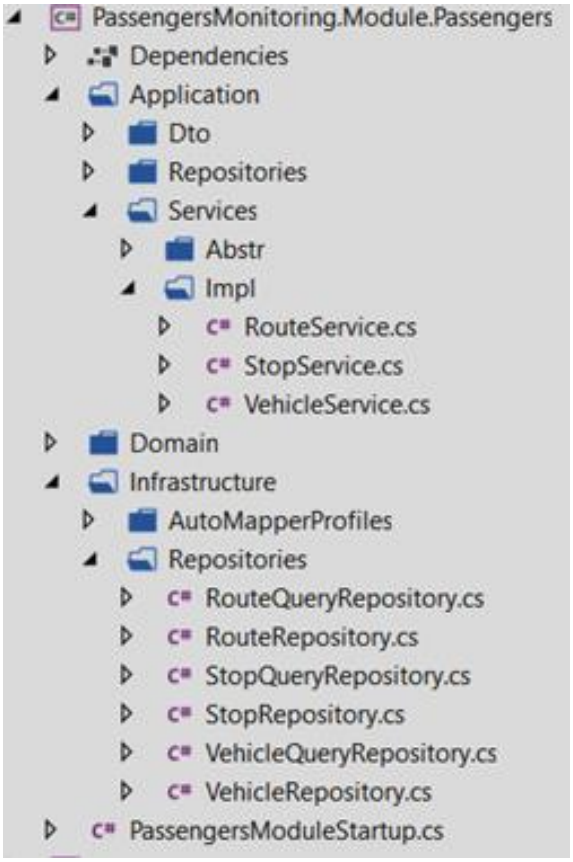

Рисунок 5.2 – Структура проєкту Module.Passengers

Тобто як ми можемо побачити з рисунку 5.2 – для зчитування даних у нас використовується QueryRepository (приклад на рисунку 5.5), а для запису даних та складних операцій над ними використовується Service (рисунок 5.6). Це і є реалізація такого підходу програмування як CQRS.

```
public class RouteDtoValidator : AbstractValidator<RouteDto>
{
    0 references
    public RouteDtoValidator()
    {
        RuleFor(x => x.OrderNumber).NotEmpty().WithMessage("Route point order number is invalid.");

        RuleFor(x => x.Location).NotEmpty().WithMessage("Route point location is invalid.");

        RuleFor(x => x.RouteNumber).NotEmpty().WithMessage("Route point number is invalid.");
    }
}
```

Рисунок 5.3 – Приклад валідації даних за допомогою «Fluent Validation»

Кожний модуль в проєкті, а саме Passenger та Identity має таку файлову структуру як показано на рисунку 5.2. У папці Application/Dto (data transfer object) зберігаються класи моделей об’єктів, за допомогою яких передаються та приймаються дані до контролерів. Використовуються вони для того, щоб об’єднати дані декількох доменних об’єктів в один з ціллю оптимізації та зменшення кількості запитів до бази даних. Також в папці Application/Dto зберігаються файли для валідації цих об’єктів за допомогою бібліотеки «Fluent Validation» (приклад на рисунку 5.3). У папці Application/Repositories зберігаються інтерфейси QueryRepository, а їх реалізація у папці Infrastructure, в ній знаходиться логіка на читання даних з БД. У папці Services знаходяться інтерфейси сервісів та їх реалізація, тобто реалізація логіки на запис даних до БД. У папці Domain знаходяться інтерфейси Repository на запис даних до БД, а їх реалізація знаходиться у папці Infrastructure. Там же в Infrastructure знаходяться файли відображення моделей DTO в доменні моделі за допомогою бібліотеки «AutoMapper» (приклад на рисунку 5.4).

```
public class StopProfile : Profile
{
    0 references
    public StopProfile()
    {
        CreateMap<Stop, StopDto>();
        CreateMap<StopDto, Stop>();

        CreateMap<CustomPoint, Point>();
        CreateMap<Point, CustomPoint>();
    }
}
```

Рисунок 5.4 – Приклад відображення за допомогою бібліотеки «AutoMapper»

```
2 references
public async Task<RouteDto> GetByIdAsync(Guid id)
{
    var route = await _context.Route
        .ProjectTo<RouteDto>(_mapper.ConfigurationProvider)
        .AsNoTracking()
        .SingleOrDefaultAsync(p => p.Id == id);

    if (route == null)
    {
        throw new Exception();
    }

    return route;
}

2 references
public async Task<RouteDto[]> GetByRouteNumberAsync(int routeNumber)
{
    var route = await _context.Route
        .Where(p => p.RouteNumber == routeNumber)
        .ProjectTo<RouteDto>(_mapper.ConfigurationProvider)
        .AsNoTracking()
        .ToArrayAsync();

    return route;
}
```

Рисунок 5.5 – Приклади методів для зчитування даних

```
2 references
public async Task<Guid> CreateRouteAsync(RouteDto routeDto)
{
    var route = await _routeRepository.CreateAsync(_mapper.Map<Route>(routeDto));
    return route.Id;
}

2 references
public async Task DeleteRouteAsync(Guid id)
{
    await _routeRepository.DeleteAsync(id);
}

2 references
public async Task UpdateRouteAsync(RouteDto routeDto)
{
    var route = await _routeRepository.GetByIdAsync(routeDto.Id.Value);
    await _routeRepository.UpdateAsync(_mapper.Map(routeDto, route));
}
```

Рисунок 5.6 – Приклади методів для запису даних

Сценарій роботи сервера на Asp.Net Core такий: застосунок очікує на HTTP-запит від клієнта і коли отримує запит, то в залежності від параметрів та інформації

отриманих від URL виконуються відповідні операції або ж то на запис або ж на читання. І далі у відповідь відповідно відправляються потрібні дані або ж повідомлення про успішну операцію у форматі JSON.

Сам Asp.Net Core застосунок розгорнуто на сервері Kestrel, а в якості проксі-сервера використовується IIS (internet information services). Тобто IIS отримує запити і перенаправляє їх застосунку, який працює на Kestrel. Така схема, коли запити йдуть не напряму на Kestrel, а проходять через IIS, дозволяє нам використовувати можливості, які є у цього веб-сервера, але відсутні у Kestrel.

Хостинг застосунку ASP.NET Core на IIS відбувається за допомогою вбудованого модуля IIS під назвою AspNetCoreModule, який налаштований таким чином, щоб перенаправляти запити на веб-сервер Kestrel. Цей модуль управляє запуском зовнішнього процесу dotnet.exe, в рамках якого хостується застосунок, і перенаправляє всі запити від IIS до цього процесу (рисунок 5.7).

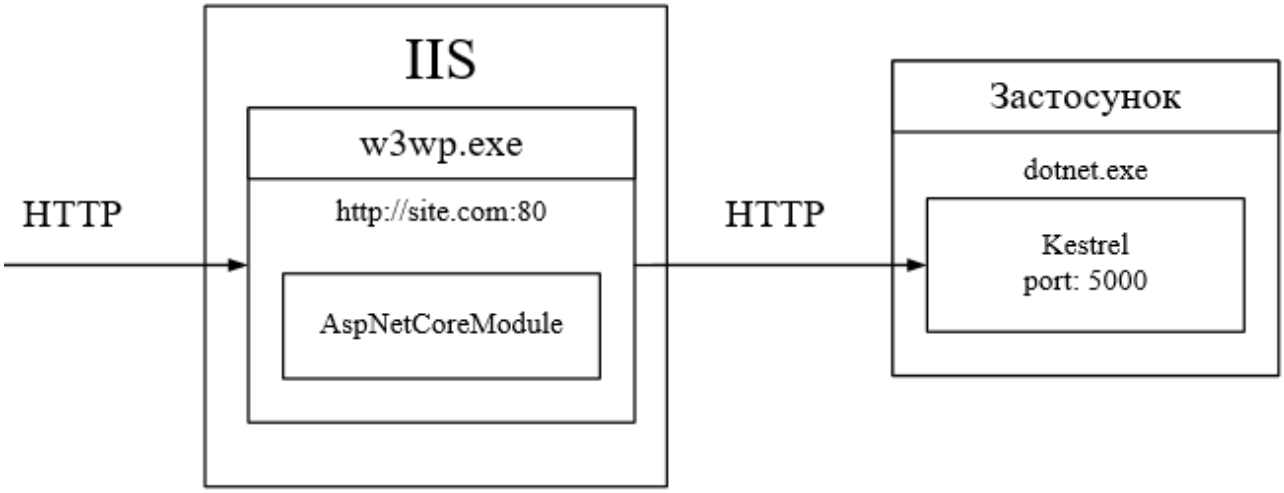

Рисунок 5.7 – Схема роботи сервера

Зазвичай застосунки написані на Asp.Net Core працюють з сервером IIS. IIS підтримує безліч різного функціоналу, має безліч можливостей з адміністрування та управління сервером і розміщеними на ньому застосунками.

Kestrel являє собою кросплатформовий веб-сервер, заснований на кросплатформенній бібліотеці асинхронного введення / виводу libuv.

Для хостингу, тобто розгортання серверу та бази даних, було обрано хмарну платформу Microsoft Azure.

Microsoft Azure — це платформа хмарних обчислень, яку можна використовувати для розробки додатків, розширення можливостей власних центрів обробки даних або в якості єдиного IT-середовища. Microsoft Azure можна швидко масштабувати відповідно до вимог.Платформа грунтується на зростаючій глобальній мережі центрів обробки даних під керуванням Майкрософт. Завдяки цьому відкривається можливість запуску застосунків із високою продуктивністю.

Спочатку на платформі розгорталась база даних PostgreSQL, для її використання встановлювалось SSL з'єднання (Secure Sockets Layer) та правила для Firewall, тобто щоб до бази даних могли під'єднатися лише клієнти з зазначеними IP-адресами (рисунок 5.8).

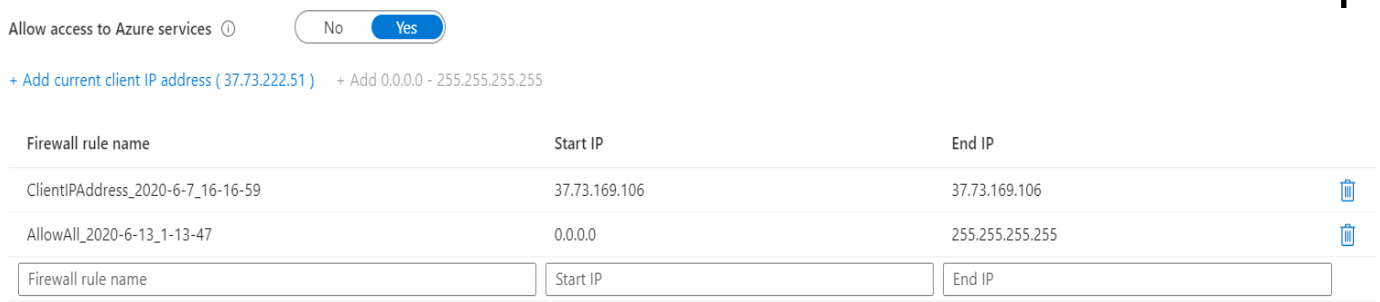

Рисунок 5.8 – Встановленні правила для Firewall

Потім у Asp.Net Core рядок підключення був змінений на відповідний для підключення до БД розгорнутої на Azure, перевірялось з'єднання та коректна передача даних та в наступному кроці сервер для застосунку теж розгортався на платформі Microsoft Azure.

Вся програмна реалізація застосунку для сервера наведена у додатку А.

5.2 Розробка бази даних сервера

Опис та призначення таблиць бази даних, що використовується в серверному проекті, зазначено у таблиці 5.1.

Таблиця 5.1 – Призначення таблиць бази даних.

Назва таблиці	Опис
stop	Таблиця зберігає дані про зупинку транспортного засобу отримані від пристрою, такі як: ідентифікатор (id), ідентифікатор транспортного засобу (vehicle_id), кількість порахованих пасажирів (passengers_number), час (timestamp) та місцеположення транспортного засобу (location) під час фіксації даних.
vehicle	Таблиця, яка зберігає у собі дані по транспотрним засобам, такі як ідентифікатор (id), модель даного транспортного засобу (vehicle_model), кількість доступних місць у ньому (sits_number) та номер маршруту по якому він їздить (route_number).
route	Таблиця містить у собі дані по маршрутам транспортних засобів, такі як: власний ідентифікатор (id), номер маршруту (route_number), місцеположення точки зупинки цього маршруту (location) та порядковий номер зупинки в даному маршруті (order_number).

Продовження таблиці 5.1.

Назва таблиці	Опис
user	Таблиця зберігає дані про користувача, а саме такі як: ідентифікатор (id), псевдонім (username) та хеш пароля (password_hash).
role	Таблиця, що відповідає за ролі користувачів. Має два поля: ідентифікатор (id) та назву ролі (name).
user_role	Єднальна таблиця, що зв’язує таблицю користувачів з таблицею ролей. Містить у собі ідентифікатори користувача (user_id) та ролі (role_id).

5.3 Структура проекту пристрою

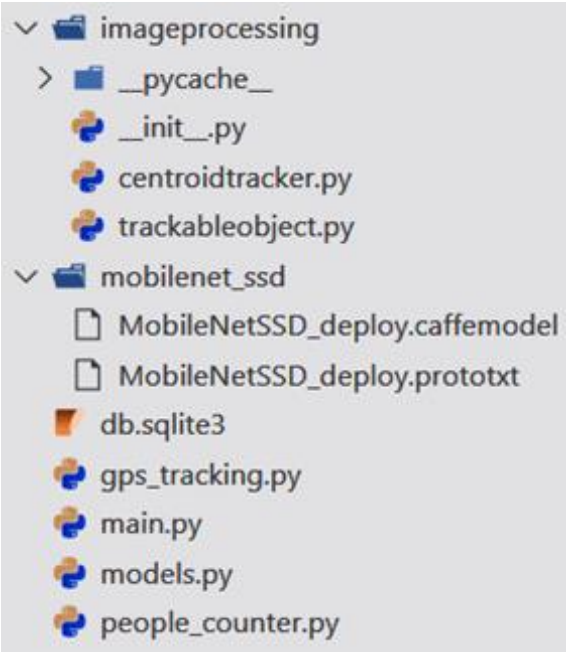

Рисунок 5.9 – Структура проекту пристрою

Структура проекту, зображена на рисунку 5.9, складається з наступних компонентів:

- imageprocessing/ – цей модуль містить алгоритм відстеження центроїдів;

- centroidtracker.py – сама реалізація алгоритму відстеження центроїдів;

- trackableobject.py – модель об'єкту яка створюється для зберігання інформації щодо відстеження та підрахунку цього об’єкту у відеопотоці;

- mobilenet_ssd/ – містить файли моделі глибокого навчання Caffe, а саме використовується MobileNet Single Shot Detector (SSD);

- db.sqlite3 – файл у якому зберігається база даних;

- gps_tracking.py – відповідає за отримання даних про місцезнаходження від gps-модуля;

-models.py – моделі для вбудованої об’єктно реляційної проекції, на базі яких створюються таблиці в базі даних;

- people_counter.py – файл у якому відбувається обробка відеопотоку, відстеження та підрахунок людей;

- __init__.py – використовується для маркування папки як пакету, щоб потім можна було імпортувати цей пакет;

- main.py – файл де знаходиться головний нескінченний цикл програми, в якому виконуються всі потрібні операції для функціонування пристрою.

Дане програмне забезпечення пристрою на мові python працює за наступним алгоритмом:

а) ввімкнення RPI;

При ввімкненні RPI відбувається запуск файлу main.py, в якому знаходиться нескінченний цикл, де перевіряється значення на піні GPIO, до якого підключений датчик дверей.

б) відкривання дверей;

Коли двері відкриваються – починається обробка відеопотоку в файлі people_counter.py за допомогою нейромережі та алгоритму відстеження центроїдів (буде описано далі), вона відбувається доки відкриті двері.

в) закриття дверей;

Коли двері зачиняються – обробка відеопотоку закінчується, з отриманих даних по кількості пасажирів, які зайшли та вийшли з ТЗ, вираховується кількість пасажирів, що знаходяться в ТЗ на даний час.

г) встановлення місцеположення;

Отримуються дані по місцеположенню з gps_tracking.py та фіксується час.

д) виконується запис в модель з файлу models.py.

е) перевіряється чи підключений пристрій до мережі Wi-Fi.

є) у випадку коли пристрій підключений до Wi-Fi, то виконуються відповідні операції;

Якщо піключенний до мережі Wi-Fi, то дані за допомогою HTTP-протоколу відправляються на сервер. Перед кожною відправкою даних до серверу перевіряється чи містить база даних записи, що ще не відправились та виконує повторне їх відправлення.

ж) у випадку коли пристрій відключений від Wi-Fi, то виконуються відповідні операції;

Якщо пристрій не підключений до мережі Wi-Fi, то дані зберігаються до локальної бази даних, яка знаходиться у файлі db.sqlite3. Перед кожним записом даних до БД перевіряється чи містяться там записи, що вже відправились і якщо такі є, то вони видаляються.

У локальній базі даних SqLite на пристрої знаходиться лише одна таблиця Stop, в якій знаходяться такі поля: кількість порахованих пасажирів (passengers_number), час (timestamp) та місцеположення транспортного засобу (location) під час фіксації даних, також в таблиці є спеціальне поле (is_sent) для того, щоб визначати чи відправлений вже даний запис до серверу чи ні.

5.4 Опис алгоритму підрахунку людей

В даному алгоритмі об'єднується концепція розпізнавання і відстеження об'єкта в єдиний алгоритм, розділяється він на дві фази:

а) розпізнавання;

Розпізнавання об'єктів - це комп’ютерна технологія, пов’язана з комп'ютерним зором та обробкою зображень, що знаходить об'єкти певного класу (таких як люди у даному проєкті) у цифрових зображеннях та відео. Приклад розпізнавання об’єктів на зображенні можна побачити на рисунку 5.10.

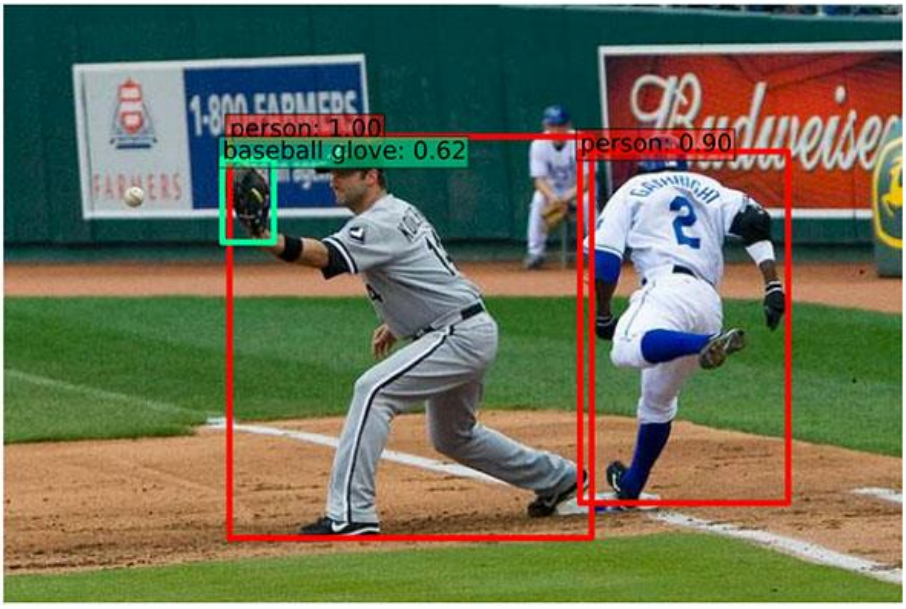

Рисунок 5.10 – Приклад розпізнавання об’єктів на зображенні виконаного за допомогою алгоритму використаного в проєкті.

На етапі розпізнавання запускається об'єктний трекер, щоб визначити, чи потрапили нові об'єкти в наше уявлення, і визначеється, чи є можливість знайти об'єкти, які були «втрачені» в час фази відстеження. Для кожного виявленого об'єкту створюється або оновлюється об'єктний трекер з новими координатами обмежувальної рамки. Оскільки об'єктний детектор більш витратний в

обчислювальному відношенні, то ця фазу запускається тільки один раз кожні N кадрів.

б) відстеження.

Відстеження об'єктів - це дисципліна в межах комп’ютерного зору, яка спрямована на відстеження об'єктів під час їх переміщення в серії відеокадрів. Об'єктами часто є люди, але можуть бути також тварини, транспортні засоби чи інші цікаві предмети, наприклад, м'яч у футбольній грі.

Коли не виконується фаза «розпізнавання», алгоритм перебуває у фазі «відстеження». Для кожного з наших виявлених об'єктів ми створюємо об'єктний трекер, який відстежує об'єкт у міру його переміщення по кадру. Наш об'єктний трекер повинен бути швидше і ефективніше детектора об'єктів. Відстеження продовжується, поки не досягає N-го кадру, а потім повторно запускається детектор об'єктів. Весь процес потім повторюється.

Перевага цього гібридного підходу полягає в тому, що можуть бути застосовані високоточні методи виявлення об'єктів без такого великого обчислювального навантаження на пристрій. Така система відстеження впроваджується, щоб створити даний лічильник людей.

Для програмної реалізації лічильника людей використовуються пакети OpenCV та dlib. З OpenCV були застосовані стандартні функції обробки зображень та комп’ютерного бачення, разом з детектором об’єктів для підрахунку людей. З пакету dlib було взято надані у ньому кореляційні фільтри, так як вони легші у використанні ніж у OpenCV.

В проекті використовується алгоритм відстеження центроїд, короткий опис якого наведено нижче:

а) Приймається набір координат обмежувальних боксів і обчислюються їх відповідні центроїди, тобто центри обмежувальних боксів (рисунок 5.10).

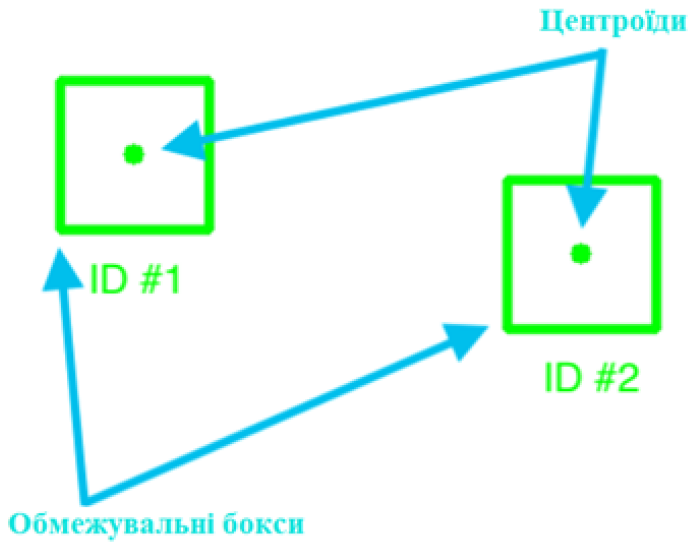

Рисунок 5.11 – Ілюстрація обмежувальних боксів та центроїд.

б) Обчислюється Евклідова відстань між будь якими новими (синій колір) та існуючими центроїдами (зелений колір) (рисунок 5.12)

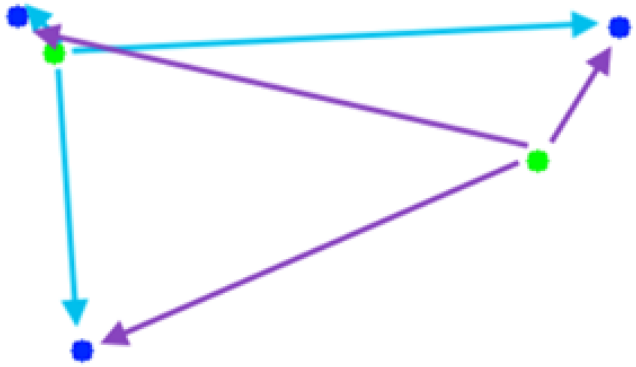

Рисунок 5.12 – Ілюстрація обчислення Евклідової відстані між центроїдами.

Алгоритм відстеження центроїд припускає, що пари центроїд з мінімальною евклідовою відстанню між собою мабуть є одним і тим самим об’єктом.

в) Як тільки ми маємо значення Евклідових відстаней – алгоритм намагається провести асоціації між парами об’єктів (рис. 5.13).

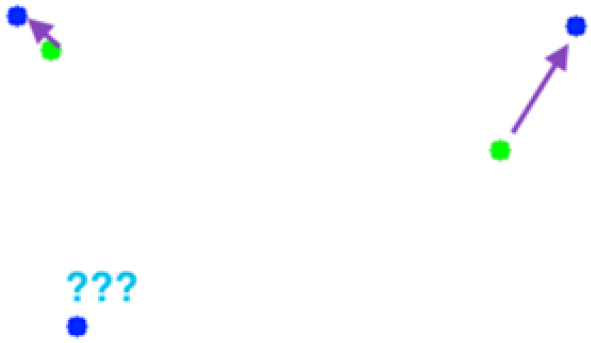

Рисунок 5.13 – Ілюстрація вибраних асоціацій об’єктів на основі Евклідових відстаней

г) Проводиться реєстрація нових об’єктів, асоціацію для яких не було знайдено на попередньому кроці (рис. 5.14). Під реєстрацією мається на увазі додавання нового об’єкту до масиву відстежуваних об’єктів, який містить id та координати центроїда нового обмежувального боксу.

Рисунок 5.14 – Реєстрація нових об’єктів.

д) Якщо об’єкт був загублений або вийшов з поля зору, то він просто видаляється. У нашому випадку об’єкт буде видалятись, коли для нього не може бути встановлена асоціація на протязі 40 послідовних кадрів відео.

Щоб відстежувати та рахувати об’єкти у відео-потоці, потрібно зберігати інформацію відносно самих об’єктів, а саме:

- id об’єкта;

- координати попередніх центроїд (щоб можна було легко визначити напрям руху об’єкта);

- був чи ні вже порахований даний об’єкт.

Тепер, з усіма описаними вище інструментами і класами, лічильник людей може бути реалізований. Опис реалізованого в проекті алгоритму наведено нижче:

а) Для ініціалізації спочатку підготовлюється SSD. Визначаються класи об’єктів, які виявляються (в нашому випадку це тільки «person») та завантажується попередньо натренована модель MobileNet SSD.

б) Визначається відео-потік з камери.

в) Інітіалізується ряд параметрів:

- W і H – розмір ширини та висоти кадрів відео;

- trackers – масив для зберігання кореляційних відстежувачів пакету dlib;

- totalFrames – загальна кількість опрацьованих кадрів;

- totalDown і totalUp – загальна кількість людей, що вийшли або ж зайшли до ТЗ;

- fps – кількість кадрів на секунду.

г) Починається циклічна обробка кадрів з потоку відео. В кінці кожної обробки перевіряється чи відкриті двері ТЗ, якщо закриті, то цикл розривається.

д) Відбувається зміна розміру кадру під встановлений в параметрах W і H, заміняються кольори, оскільки для dlib потрібне rgb зображення.

е) Визначаються люди в кадрі використовуючи SSD. Спочатку статус обробки ініціалізується як «waiting» і при початку обробки змінюється на «detecting». Взагалі можливі такі статуси:

- waiting – алгоритм чекає на початок обробки та виявлення людей;

- detecting – відбувається визначення наявності людей в кадрі (один раз на 30 кадрів);

- tracking – люди відстежуються в кадрі і підраховуються totalUp і totalDown (на протязі 29 з 30 кадрів, що залишились).

є) Тобто щоб уникнути обробки зображення для визначення людей на кожному кадрі відео та пришвидшити обробку – пропускається 30 кадрів. Для цього на кожному кадрі відбувається перевірка чи це 30-й кадр чи ні.

ж) Проводиться виявлення об’єктів передавши blob отриманий від зображення через мережу нейронної моделі та отримуємо масив detections. Вибираємо з цього масиву об’єкти класу «person» відкинувши результати з низькою ймовірністю і ті об’єкти що не належать до цього класу. З цих об’єктів отримуємо їх обмежувальні бокси. Далі пункти з 4 по 8 відбуваються в циклі через кожні 30 кадрів.

з) В інших 29 кадрах відео відстежуємо ці об’єкти за допомогою кореляційного відстежувача пакету dlib.

и) Перебираються доступні відстежувачі, статус змінюється на «tracking». Витягуються координати для об’єктів які відстежуються.

й) Визначається умовна горизонтальна лінія на відео, для визначення людей які її перетинають. Далі відповідно до того вверх чи вниз рухається особа – додається 1 до totalUp або totalDown відповідно. Для визначення напрямку руху беруться y-координати усіх попередніх центроїд об’єкта і від позиції центроїда на даний час віднімаємо їх середнє значення. Таким чином в залежності від знаку результату визначається напрям руху.

к) І в кінці даний об’єкт зберігається до trackableObjects dictionary, щоб можна було взяти цей об’єкт для його встановлення в наступному кадрі.

л) Цикл починається з початку.

Програмну реалізацію описаного проекту пристрою наведено у додатку Б.

5.5 Опис поєднаних методів «Single Shot Detectors» і «MobileNets»

5.5.1 Метод «Single Shot Detectors»

Що стосується розпізнавання на основі «deep learning», то існує три основні методи виявлення об'єктів:

- Швидкий R-CNNs;
- You Only Look Once (YOLO);
- Single Shot Detectors (SSDs).

Швидкий R-CNN, ймовірно, є найбільш відомим методом виявлення об'єктів за допомогою «deep learning», проте метод може бути важкий для розуміння, в реалізації та складною для тренування.

Окрім того, навіть при «швидшій» реалізації R-CNN алгоритм є досить повільним, приблизно 7 FPS.

Якщо потрібна чиста швидкість, то рекомендується використовувати YOLO, оскільки цей алгоритм набагато швидший, здатний обробляти 40-90 FPS на GPU Titan X. Супер швидкий варіант YOLO може отримати навіть 155 FPS. Але проблема YOLO полягає в тому, що точність даного методу залишає бажати кращого.

SSD (Single Shot Detectors), спочатку розроблений Google – це баланс між ними. Алгоритм більш простий (можна стверджувати, що краще пояснюється), ніж швидкий R-CNN

5.5.2 Метод «MobileNets»

При побудові мереж розпізнавання об'єктів зазвичай використовується існуюча мережева архітектура, така як VGG або ResNet, а потім вона використовується всередині конвеєра розпізнавання об'єктів. Проблема полягає в тому, що ці мережеві архітектури можуть бути дуже великими в порядку 200-500 МБ.

Але такі мережеві архітектури непридатні для обмежених ресурсів пристрою, як Raspberry Pi у нашому випадку, через їх розмір і кількість обчислень.

Натомість в проєкті використовується «MobileNets» - ще одна розробка Google. Ці мережі називаються «MobileNets», оскільки вони розроблені для пристроїв з обмеженими ресурсами. MobileNets відрізняються від традиційних CNN через використання згортки, що розділяється по глибині. На рисунку 5.15 зліва зображений блок звичайної згорткової мережі з нормалізацією пакета (BN) та ReLU, а справа - базовий блок MobileNet з глибоко відокремлюваною згорткою з поглибленим і точковим шарами з подальшою нормалізацією пакету та ReLU.

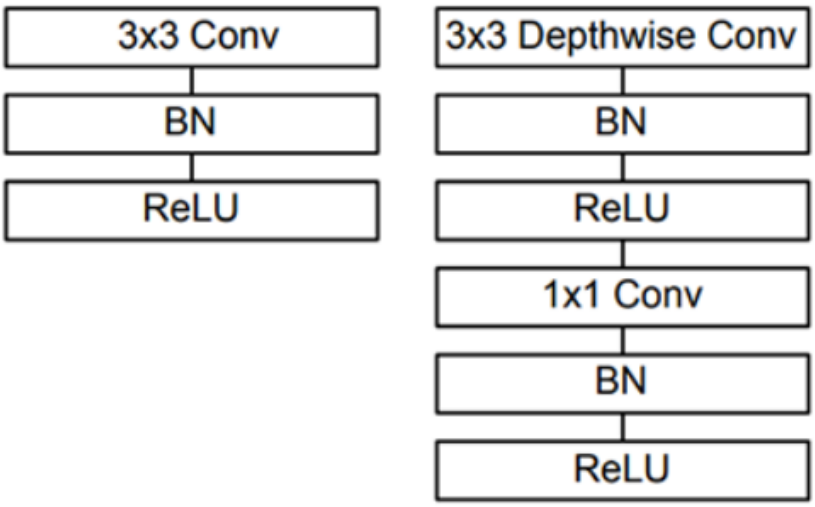

Рисунок 5.15 – Блок звичайної згорткової мережі та базовий блок MobileNet [18]

Загальна ідея, що знаходиться за глибоко відокремленою згорткою - це розділити згортку на два етапи:

- 3x3 глибока згортка;

- 1x1 точкова згортка.

Це дозволяє нам фактично зменшити кількість параметрів у нашій мережі. Але проблема полягає в тому, що страждає точність – MobileNets зазвичай не настільки точні, як великі подібні мережі, але вони набагато ефективніші.

5.5.3 Поєднання «Single Shot Detectors» і «MobileNets» для швидкого розпізнавання об’єктів на основі «deep learning»

При поєднанні такої архітектури як «MobileNets» і «Single Shot Detectors» (SSD), ми досягаємо швидкого, ефективного методу «deep learning» для розпізнавання об'єктів.

Модель, яку ми будемо використовувати в даному проєкті – це «Caffe» версія оригінальної «TensorFlow» реалізації.

«MobileNet SSD» був спочатку натренований на наборі даних «COCO» (Common Objects in Context), а потім налаштований за допомогою «PASCAL VOC», досягши 72.7% mAP (середня точність).

5.6 Висновки до розділу

У даному розділі було описано структуру розробленої серверної частини. А саме визначено архітектуру та підходи за якими будується даний проєкт, описаний функціонал кожного файлу. Було також описано шлях розгортання сервера для даного застосунку в мережі інтернет. Далі було розписано структуру побудови проєкта на пристрої. Тобто було наведено покрокове виконання алгоритму підрахунку людей, алгоритму відстеження та розпізнавання об’єктів, надано вичерпну інформацію про компоненти системи, їх взаємодію та принцип роботи.

6 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування програмного забезпечення - це процес, що дозволяє оцінити функціональність програмного забезпечення з метою встановити, чи відповідає розроблене програмне забезпечення заданим вимогам чи ні, та визначити дефекти, щоб забезпечити відсутність їх відсутність та якість розробленого продукту.

Існують різні рівні тестування:

- тестування модулів;
- інтеграційне тестування;
- тестування системи.

Тестування модулів – проводиться для перевірки правильності роботи окремих модулів вихідного коду.

Інтеграційне тестування – це процес випробування на підключення або передачу даних між парою перевірених модулів.

Тестування системи (тестування від кінця до кінця) – це тестування повністю інтегрованої програми, що також називається тестуванням сценарію. Виконується для забезпечення коректної роботи програмного забезпечення у всіх призначених цільових системах.

В розробленій системі – тестування проводиться тільки для серверного застосунку на Asp.Net Core. За це відповідає проєкт PassengersMonitoring.Test. Для написання тестів використовується така бібліотека, як NUnit. Для емуляції функціональності та створення мок-об’єктів – використовується бібліотека Moq. В нашому випадку були створені тільки модульні тести і проводяться вони для тестування функціональності контролерів і сервісів. Приклад модульного тесту в якому проводиться тест на видалення неіснуючого об'єкту через RouteController можна побачити на рисунку 6.1.

```
[Test]
0 references
public async Task Delete_route_should_return_bad_request_for_missing_id()
{
    var routeServiceMock = new Mock<IRouteService>();
    var routeQueryRepositoryMock = new Mock<IRouteQueryRepository>();
    var controller = new RouteContoller(routeQueryRepositoryMock.Object, routeServiceMock.Object);
    var id = Guid.Empty;

    var result = await controller.DeleteRouteById(id);

    Assert.IsInstanceOf<BadRequestResult>(result);
}
```

Рисунок 6.1 – Приклад модульного тесту

Список усіх тестів, що проводяться в нашому проєкті ми можемо побачити на таблиці 6.1.

Таблиця 6.1 – Список усіх тестів, що проводяться в проєкті.

Назва	Опис
Створення Stop в StopController.	При успішному виконанні повинен повернути код HTTP 200 OK.
Отримати Stop по id в StopController.	Повинен повернути об’єкт типу Stop.
Отримати Stop по неіснуючому id в StopController.	Повинен повернути помилку BadRequestResult.
Створення Route в RouteController.	При успішному виконанні повинен повернути код HTTP 200 OK.
Видалення Route по id в RouteController.	При успішному виконанні повинен повернути код HTTP 200 OK.
Видалення Route по неіснуючому id в RouteController.	Повинен повернути помилку BadRequestResult.
Створення User в UserIdentityController	При успішному виконанні повинен повернути токен.

Назва	Опис
Створення Vehicle в RouteController.	При успішному виконанні повинен повернути код HTTP 200 OK.
Видалення Vehicle по id в RouteController.	При успішному виконанні повинен повернути код HTTP 200 OK.
Отримати Vehicle по id в VehicleController.	Повинен повернути об’єкт типу Stop.
Оновити існуючий Vehicle по вхідному об’єкту Vehicle в VehicleController.	При успішному виконанні повинен повернути код HTTP 200 OK.

6.1 Висновки до розділу

В даному розділі було описано які існують тестові рівні та яким чином відбувається тестування в серверній частині проєкту. Було наведено приклад тестового випадку на рисунку. І також було наведено список реалізованих в проєкті тестових випадків у вигляді таблиці.

ВИСНОВКИ

При виконанні даного дипломного проекту було створено автоматичну систему моніторингу та аналізу кількості пасажирів у транспортних засобах. А саме серверну частину на основі фреймфорку Asp.Net Core та сам проект для пристрою.

Спочатку було проведено аналіз предметної області розроблюваної системи. Було визначено призначення, цілі та задачі розробки, а також було визначено основні особливості майбутньої системи. Далі були проаналізовані існуючі рішення, було створено детальний опис кожного з них. Були розглянуті та проаналізовані переваги та недоліки аналогів. На основі отриманих даних були розроблені вимоги до системи. Було проведено вибір технологій розробки та компонентів пристрою з обґрунтуванням цього вибору. Далі, на основі вимог, визначених раніше, за допомогою обраних технологій було описано та розроблено систему.

Розроблене програмне забезпечення готове до застосування та відповідає усім визначеним вимогам. Також, існує можливість розширення функціоналу за допомогою модулів, які можна додати потім.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. David M. Wheeler, Damilare D. Fagbemi. The IoT Architect's Guide to Attainable Security and Privacy. 2020. 300 p.

2. K. Terada, D. Yoshida, S. Oe, J. Yamaguchi. A method of counting the passing people by using the stereo images. 1999. 338–342 p.

3. D. Beymer. Person counting using stereo, 2000. 127–133 p.

4. K. Hashimoto, K. Morinaka, N. Yoshiike, S. Kawaguchi, C. Matsueda. People count system using multi-sensing application. 1997. 1291–1294 p.

5. A. T. A. T. C. S. R. G. Vernazza. Long-memory matching of interacting complex objects from real image sequences. 1996. 120 p.

6. A. Shio, J. Sklansky. Segmentation of people in motion. 1991. 325–332 p.

7. G. Sexton, X. Zhang, G. Redpath, D. Greaves. Advances in automated pedes-trian counting. 1995. 106–110 p.

8. J. Segen. A camera-based system for tracking people in real time. 1996. 63–67 p.

9. Thanarat Horprasert , David Harwood , Larry S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. 1999. 50 p.

10. J.-S. Hu, T.-M. Su, S.-C. Jeng. Robust background subtraction with shadow and highlight removal for indoor surveillance. 2006. 4545–4550 p.

11. I. Haritaoglu, M. Flickner. Detection and tracking of shopping groups instores. 2001. 120 p.

12. A. B. Chan, Z.-S. J. Liang, N. Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. 2008. 1-7 p.

13. A. B. Chan, N. Vasconcelos. Modeling, clustering, and segmenting video with mixtures of dynamic textures. 2008. 909–926 p.

14. L. F. Teixeira, L. Corte-Real. Video object matching across multiple independent views using local descriptors and adaptive learning. 2009. 157–167 p.

15. Mark Lutz. Programming Python: Powerful Object-Oriented Programming. O`Reilly Media, 2011.

16. Sibsankar Haldar. SQLite Database System Design and Implementation. Sibsankar Haldar, 2015.

17. Features Of SQLite. URL: https://www.sqlite.org/features.html (дата звернення: 23.05.2020).

18. MobileNet: меньше, быстрее, точнее. URL: https://habr.com/ru/post/352804/ (дата звернення: 14.05.2020).